Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)
Навчально-науковий інститут комп'ютерних наук
та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))
Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

Магістерська кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему: : «Інтелектуальна система «Спортивний тренер» засобами react»

Виконав: студент VI курсу групи КН-62м
Спеціальності:
122 "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)

Місьо А.Р.
(прізвище та ініціали)

Керівник Яцишин С.І.
(прізвище та ініціали)

Рецензент Фаєд Л.О.
(прізвище та ініціали)

Львів – 2025

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ комп`ютерних наук та інформаційних технологій
Кафедра комп`ютерних наук
Рівень вищої освіти другий (магістерський)
Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук

_____ Борецька І.Б.

"10" грудня 2025 року

З А В Д А Н Н Я

НА МАГІСТЕРЬСКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Місьо Андрій Романович
(прізвище, ім'я, по батькові)

1. Тема роботи: Інтелектуальна система «Спортивний тренер» засобами react
Керівник роботи *Яцишин С. І., кандидат технічних наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "29" квітня 2025 року №С-288
2. Термін подання студентом роботи 10.12.2025
3. Вихідні дані до роботи: створення інтелектуальної системи «Спортивний тренер» засобами react
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Розділ 1. Стан проблемної області
Розділ 2. Інформаційне забезпечення
Розділ 3. Математичне забезпечення
Розділ 4. Програмне забезпечення
Розділ 5. Розроблення стартап-проєкту
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Підготовка матеріалу до доповіді
6. Дата видачі завдання 01.05.2025

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання стапів роботи	Примітка
1.	Огляд літератури згідно досліджуваної теми. Збір необхідних матеріалів.	01.05.25-15.06.25	Виконано
2.	Постановка задачі і її формалізація	16.06.25-30.06.25	Виконано
3.	Виконання вхідного етапу технології	02.07.25 – 29.07.25	Виконано
4.	Реалізація головних алгоритмів проекту	30.08.25 – 30.09.25	Виконано
5.	Виконання етапу відлагодження проєкту	01.10.25 – 31.10.25	Виконано
6.	Виконання етапу впровадження та випуску бета-версії.	02.11.25 – 16.11.25	Виконано
7.	Оформлення записки до дипломного проекту.	17.11.25 – 07.12.25	Виконано

Студент _____ Місьо А.Р.
(підпис) (прізвище та ініціали)

Керівники роботи _____ Яцишин С.І.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Магістерська робота містить 87 с., 27 рис., 20 джерел.

Розроблено інтелектуальну веб-систему «Спортивний тренер» для персоналізації тренувань та контролю техніки вправ засобами комп'ютерного зору. Система дозволяє формувати та відстежувати індивідуальні плани користувачів. Аналіз рухів у реальному часі реалізовано за допомогою MediaPipe Pose. Зберігання даних про прогрес забезпечує платформа Firebase (Auth, Firestore). Клієнтська частина створена на фреймворку React.

Ключові слова: React, MediaPipe Pose, Firebase, комп'ютерний зір, оцінка пози, інтелектуальна система, веб-додаток.

ABSTRACT

The thesis consists of 87 pages, 27 illustrations, and 20 references.

The "Sports Trainer" intelligent web system was developed for personalized training and technique monitoring using computer vision. The system allows creating and tracking individual user plans. Real-time movement analysis is implemented using MediaPipe Pose. Progress data storage is handled by the Firebase platform (Auth, Firestore). The client-side is built with the React framework.

Keywords: React, MediaPipe Pose, Firebase, computer vision, pose estimation, intelligent system, web application.

ТЕХНІЧНЕ ЗАВДАННЯ

Відповідно до встановлених вимог технічного завдання, необхідно розробити Інтелектуальну систему «Спортивний тренер». Ця система є сучасним веб-сервісом, призначеним для автоматизації персонального тренувального процесу та об'єктивного контролю за якістю виконання фізичних вправ.

Користувачам надається можливість отримати індивідуалізований тренувальний план, який динамічно коригується на основі їхніх цілей, фізичних параметрів та фактичного прогресу. Реалізація функціоналу ґрунтується на хмарній платформі Firebase для надійної автентифікації користувачів та зберігання великих обсягів даних про їхні тренування.

Ключовим елементом системи є інтелектуальний модуль, який використовує бібліотеку MediaPipe Pose для захоплення та аналізу рухів тіла людини у реальному часі через вебкамеру. Цей модуль дозволяє автоматично розпізнавати тип вправи (наприклад, присідання, статичні стійки), точно підраховувати кількість повторень або час утримання пози, а також оцінювати правильність техніки виконання.

Система повинна передбачати функцію візуалізації прогресу, де користувачі можуть відстежувати свою динаміку, включаючи графіки зміни ваги та статистики виконаних вправ. Головною метою розробки є гарантування простоти, зручності та інтуїтивності інтерфейсу, реалізованого на React, що забезпечує високу інтерактивність та адаптивність для кінцевих користувачів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ТЗ — Технічне завдання

AI — Artificial Intelligence (Штучний інтелект)

ML — Machine Learning (Машинне навчання)

CV — Computer Vision (Комп'ютерний зір)

HPE — Human Pose Estimation (Оцінка пози людини)

UI — User Interface (Користувацький інтерфейс)

UX — User Experience (Користувацький досвід)

BaaS — Backend as a Service (Бекенд як послуга)

SDK — Software Development Kit (Комплект засобів розробки)

React — JavaScript-бібліотека для розробки користувацьких інтерфейсів

MediaPipe — Фреймворк Google для створення конвеєрів машинного навчання та комп'ютерного зору

Firebase — Хмарна платформа Google для розробки мобільних та веб-додатків

Firestore — NoSQL-база даних у складі Firebase

API — Application Programming Interface (Інтерфейс прикладного програмування)

HTML — HyperText Markup Language (Мова розмітки гіпертексту)

CSS — Cascading Style Sheets (Каскадні таблиці стилів)

JSON — JavaScript Object Notation (Формат обміну даними)

ВСТУП

В умовах стрімкої діджиталізації суспільства та зростання популярності здорового способу життя, попит на персоналізовані та доступні фітнес-рішення невпинно зростає. Традиційні методи тренувань часто вимагають присутності кваліфікованого тренера, що є не завжди доступним або економічно вигідним. Існуючі мобільні додатки та веб-платформи здебільшого пропонують статичні плани тренувань і покладаються на самооцінку користувача, не забезпечуючи об’єктивного контролю за якістю та технікою виконання вправ. Неправильна техніка, у свою чергу, знижує ефективність тренувань та суттєво підвищує ризик травматизму. Вирішення цієї проблеми лежить у площині інтеграції технологій комп'ютерного зору (CV) та штучного інтелекту (AI) в онлайн-тренінг. Розробка інтелектуальної системи, здатної аналізувати положення тіла користувача в режимі реального часу та надавати оперативний зворотний зв’язок, є критично актуальною. Дана магістерська робота присвячена розробці такої системи – «Спортивний тренер». Використання сучасного стеку технологій, зокрема фреймворку React для створення високоінтерактивного клієнтського інтерфейсу, бібліотеки MediaPipe Pose для точного та швидкого визначення пози людини (Human Pose Estimation, HPE) через вебкамеру, та хмарної платформи Firebase для масштабованого управління даними та автентифікацією, дозволяє створити надійне, ефективне та доступне рішення. Актуальність роботи визначається необхідністю створення нових, технологічно просунутих інструментів для персоналізації та об’єктивного контролю у сфері фізичного виховання та спорту.

Мета роботи полягає у розробці теоретичних засад та практичній реалізації інтелектуальної веб-системи «Спортивний тренер» для формування, коригування та контролю індивідуальних тренувальних планів на основі об’єктивного аналізу даних користувача та його рухів у реальному часі. Для досягнення поставленої мети визначено такі завдання дослідження: 1. Проаналізувати сучасні методи комп'ютерного зору та підходи до оцінки пози людини для автоматизації тренувального процесу. 2. Обґрунтувати вибір технологічного стеку: React, Firebase (Auth, Firestore),

та бібліотеки MediaPipe Pose, з урахуванням вимог до швидкодії та масштабованості. 3. Розробити архітектуру системи, орієнтовану на хмарні сервіси Firebase, та спроєктувати модель даних для ефективного зберігання спортивних показників та журналів тренувань. 4. Створити та деталізувати алгоритми аналізу рухів (динамічних та статичних вправ) на основі координат ключових точок, що надаються MediaPipe. 5. Реалізувати інтерактивний клієнтський додаток засобами React та забезпечити його інтеграцію з Firebase для управління даними та ідентифікації користувачів. 6. Розробити модуль візуалізації прогресу (з використанням Chart.js), включаючи графіки динаміки виконання вправ та зміни фізичних показників (наприклад, ваги). 7. Провести експериментальне тестування розробленої системи та оцінити її ефективність і точність розпізнавання рухів.

Об’єктом дослідження є процеси персоналізації, автоматизації та об'єктивного контролю техніки у сфері фізичних тренувань. Предметом дослідження є моделі, методи та програмне забезпечення для побудови інтелектуальної системи «Спортивний тренер» на базі бібліотеки React та хмарної платформи Firebase.

Наукова новизна отриманих результатів полягає у: удосконаленні методології об'єктивного контролю техніки виконання фізичних вправ шляхом інтеграції моделі MediaPipe Pose безпосередньо у клієнтський веб-додаток, що мінімізує затримки та забезпечує високоточний аналіз рухів у реальному часі; розробці алгоритмів оцінки якості виконання вправ (таких як присідання, планка) на основі геометричного аналізу ключових точок тіла, що дозволяє автоматично виявляти типові помилки в техніці; обґрунтуванні та реалізації архітектурного рішення, яке поєднує високопродуктивний інтерфейс на React із масштабованими хмарними сервісами Firebase Firestore та Authentication для створення адаптивної системи управління тренувальним процесом. Практичне значення отриманих результатів: створено функціональний прототип веб-системи, який може бути використаний як інструмент для самостійних тренувань, забезпечуючи корекцію рухів та запобігаючи травмам; розроблена система є гнучкою та може бути легко адаптована для різних видів фізичної активності (йога, пілатес, силові тренування); запропоновані архітектурні та

алгоритмічні рішення можуть бути використані при розробці комерційних продуктів для віддаленого фітнес-тренінгу та телемедицини.

Апробація та структура роботи. Основні положення та результати магістерської роботи були апробовані у процесі розробки програмного забезпечення. Магістерська робота складається з вступу, трьох розділів, висновків, переліку використаних джерел та додатків. Розділ 1 присвячено аналізу сучасного стану фітнес-технологій, методів комп'ютерного зору (HPE) та обґрунтуванню вибору технологічного стеку (React, MediaPipe, Firebase). Розділ 2 описує розробку архітектури, проєктування моделі даних у Firebase та деталізацію алгоритмів інтелектуального модуля для аналізу рухів. Розділ 3 містить опис програмної реалізації клієнтської частини на React, інтеграцію з MediaPipe та Firebase, а також результати експериментальної апробації системи.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Аналіз еволюції фітнес-технологій та роль персоналізованого тренінгу

Еволюція технологій у сфері фізичного виховання та фітнесу відображає загальні тенденції діджиталізації суспільства та розвиток методів обробки даних. Цей шлях можна чітко розділити на кілька ключових етапів, кожен з яких привносив нові інструменти для моніторингу та управління тренувальним процесом.

Етап пасивного моніторингу та механічних систем (До 2000-х років). Початковий етап характеризувався домінуванням традиційних механічних тренажерів та перших електронних пристроїв. Основний акцент робився на кількісному вимірюванні базових параметрів. До цих пристроїв належали прості пульсометри, які вимірювали частоту серцевих скорочень, та перші крокоміри. Ці інструменти забезпечували лише пасивний збір даних — інформація фіксувалася, але не аналізувалася інтелектуально. Основною проблемою цього етапу була відсутність зв'язку між зібраними біометричними даними та якістю самого тренувального процесу, а також повна відсутність можливостей для індивідуалізації.

Етап цифрової автоматизації та мобільних платформ (2000–2015 роки). Зі стрімким поширенням мобільних пристроїв та доступом до мережі Інтернет розпочалася ера цифрової автоматизації. Поява смартфонів дозволила розробляти фітнес-додатки, які агрегували дані з GPS, акселерометрів та перших носійних пристроїв (фітнес-браслетів). Користувачам пропонувалися великі бібліотеки статичних відео-інструкцій та фіксовані тренувальні плани. Хмарні технології, що розвивалися паралельно, дозволили зберігати історію тренувань централізовано. Ключові досягнення та обмеження етапу: Досягненням було створення зручного трекінгу, соціалізація фітнесу, автоматичне планування за розкладом. Мобільні додатки стали основним каналом взаємодії з користувачем. Обмеженням була переважна універсальність планів (one-size-fits-all), без глибокої адаптації до індивідуальної фізіології. Найважливішим недоліком була відсутність об’єктивного контролю техніки. Користувач мусив самостійно оцінювати своє положення, що часто призводило до

неефективності, застою у прогресі та, що найважливіше, до підвищеного ризику отримання травм через неправильну біомеханіку рухів.

Перехід до інтелектуальних та адаптивних систем (З 2015 року). Сучасний етап еволюції фітнес-технологій нерозривно пов'язаний із інтеграцією штучного інтелекту (AI), машинного навчання (ML) та комп'ютерного зору (CV). Цей перехід відрізняється фокусом на адаптивності та об'єктивному контролі. Серцем сучасних систем, таких як розроблювана «Спортивний тренер», є глибока персоналізація. Вона виходить далеко за межі простого вибору цілі ("схуднути" або "набрати м'язи"). Персоналізований тренінг вимагає безперервного аналізу великого масиву даних: 1) Початкові дані: Антропометрія, рівень підготовки, історія тренувань та травм (зберігаються у Firebase Firestore). 2) Динамічні дані: Реакція організму на навантаження, частота виконання, швидкість відновлення, суб'єктивна оцінка втоми. На основі цього аналізу алгоритми машинного навчання можуть динамічно коригувати план тренувань, змінюючи інтенсивність, обсяг або тип вправ, оптимізуючи прогрес і запобігаючи перетренованості. Персоналізація є критично важливою, оскільки вона гарантує, що тренувальна програма є максимально ефективною та безпечною саме для конкретного користувача.

Об'єктивний контроль за допомогою Комп'ютерного зору. Найбільш значущим проривом стало впровадження систем Оцінки пози людини (Human Pose Estimation, HPE). Ці системи дозволяють замінити фізичну присутність тренера в плані моніторингу техніки. Використовуючи вебкамеру, модель HPE (наприклад, MediaPipe Pose) може ідентифікувати 33 ключові точки тіла (суглоби) і перетворити їх на дво- або тривимірні координати. На основі цих координат можна реалізувати алгоритми, що: вимірюють кути суглобів (наприклад, кут коліна під час присідання) для оцінки глибини та правильності руху; відстежують траєкторію руху всього тіла або окремих кінцівок для контролю рівноваги та стійкості (критично важливо для асан або статичних вправ); автоматично підраховують кількість виконаних повторень, фіксуючи перехід між фазами вправи (наприклад, опускання і підйом при присіданні). Такий зворотний зв'язок у реальному часі є фундаментальною перевагою інтелектуальних систем. Він не тільки забезпечує безпеку, запобігаючи травмам

через хибну техніку (наприклад, округлення спини під час тяги), але й підвищує ефективність, гарантуючи, що м'язи працюють у потрібному діапазоні.

Контекст даного дослідження. Розроблена система «Спортивний тренер» стоїть на стику цих двох прогресивних напрямків: вона використовує React для створення надійного фронтенду та інтеграції вебкамери, MediaPipe Pose для високоточного об’єктивного контролю, і Firebase для забезпечення масштабованості та глибокої персоналізації. Це рішення є актуальним внеском у розвиток фітнес-технологій, оскільки воно об’єднує передові методи CV та BaaS-архітектури, пропонуючи користувачеві доступний і науково обґрунтований інструмент для досягнення своїх фітнес-цілей.

1.2. Огляд методів комп'ютерного зору в реальному часі для оцінки рухів людини (Pose Estimation)

Успішна реалізація інтелектуальної системи «Спортивний тренер» безпосередньо залежить від ефективності модуля комп'ютерного зору, здатного точно та швидко визначати позу людини (Human Pose Estimation, HPE) у відеопотоці з вебкамери. HPE — це фундаментальна задача, що полягає у визначенні положення ключових суглобів та частин тіла (так званих landmarks) та їхніх зв’язків. Історично методи HPE еволюціонували від підходів, що ґрунтувалися на графічних моделях, до сучасних систем, які використовують глибокі згорткові нейронні мережі (CNN) [2, 6], оптимізовані для виконання обчислень у реальному часі.

Існують два основні класи методів оцінки пози. Перший, Top-Down (від загального до деталей), спочатку використовує детектор об’єктів для виявлення людини в кадрі, а потім, у межах знайденого обмежувального прямокутника, виконує оцінку пози. Цей метод зазвичай пропонує високу точність, але його обчислювальна складність є прямо пропорційною кількості людей у кадрі, що робить його менш оптимальним для високопродуктивного веб-застосунку. Другий підхід, Bottom-Up (від деталей до загального), спочатку виявляє всі ключові точки в кадрі незалежно від того, якій особі вони належать, а потім використовує методи асоціації, щоб

згрупувати ці точки з конкретними людьми. Він є більш ефективним у сценах з великою кількістю людей, але може мати нижчу точність у визначенні індивідуальних точок.

Для інтеграції у веб-додаток, де обчислення часто відбуваються на клієнтському пристрої (браузері), критично важливим є вибір рішення, що забезпечує оптимальний баланс між точністю та швидкістю. Серед ключових рішень, OpenPose, хоч і відомий своєю високою точністю [1], часто має занадто високі обчислювальні вимоги для стандартного браузера. MoveNet, розроблений Google, є надзвичайно швидкою моделлю [11], оптимізованою для мобільного та веб-середовища, проте може поступатися в точності при складному біомеханічному аналізі.

Для реалізації інтелектуальної системи «Спортивний тренер» було обрано фреймворк MediaPipe Pose від Google [8]. Його вибір обґрунтований комплексним, конвеєрним підходом, що забезпечує як високу точність, так і необхідну продуктивність [7]. Архітектура MediaPipe поєднує детектор людини та спеціалізовану модель відстеження (Landmark Tracking). Це унікальне поєднання значно підвищує швидкість і стабільність результату порівняно з простим повторним детектором на кожному кадрі. Найголовнішою перевагою MediaPipe є його здатність оцінювати 33 ключові точки тіла (landmarks), надаючи їхні координати у тривимірному (3D) просторі. Це дозволяє точно розрахувати кути між суглобами (наприклад, кут коліна або стегна) з мінімальною похибкою, що є необхідною основою для всіх алгоритмів оцінки техніки та підрахунку повторень. Модель розроблена для виконання у браузері, використовуючи технології WebAssembly та WebGL/WebGPU для апаратного прискорення, що гарантує обробку відеопотоку у реальному часі (25–30 кадрів на секунду) без значного навантаження на мережу чи серверний бекенд. Таким чином, MediaPipe Pose є найбільш оптимальним інструментом для реалізації зворотного зв'язку у реальному часі щодо техніки виконання вправ, що є центральною функцією розроблюваної інтелектуальної системи.

1.3. Порівняльний аналіз архітектурних рішень для веб-додатків. Обґрунтування вибору технологічного стеку: React та Firebase

Вибір архітектурного рішення та основного технологічного стеку є критично важливим для успішної реалізації високоінтерактивної, масштабованої та відмовостійкої веб-системи, такої як «Спортивний тренер». Оскільки ключовим елементом системи є обробка відео в реальному часі та оперативне оновлення даних, архітектура повинна забезпечувати високу швидкість відгуку та легкість управління станом програми.

На початковому етапі було проаналізовано три основні групи архітектурних рішень: традиційний клієнт-серверний підхід (з власним бекендом), рішення на базі BaaS (Backend as a Service) та архітектура Single Page Application (SPA). Для фронтенду було розглянуто провідні JavaScript-фреймворки, такі як Angular, Vue.js та React. Angular є комплексним, потужним фреймворком, добре підходить для великих корпоративних програм, але його "жорстка" структура може бути надмірною для проєкту середнього розміру з акцентом на швидку інтерактивність. Vue.js пропонує простоту та гнучкість, але React виділяється найбільшою екосистемою, зрілістю та, що найважливіше, ідеальною інтеграцією з інструментами для маніпуляцій DOM, які необхідні для високочастотного оновлення інтерфейсу на основі даних, отриманих від MediaPipe Pose. Декларативний підхід React, його компонентна модель та механізм Virtual DOM [9] забезпечують ефективну реактивність, що є необхідною умовою для відображення ключових точок тіла (landmarks) та миттєвого зворотного зв'язку у реальному часі.

Що стосується бекенду, рішення про вибір архітектури BaaS на основі Firebase було обґрунтовано потребами проєкту в швидкості розробки [16], масштабованості та синхронізації даних у реальному часі. Традиційний підхід вимагав би розробки власного RESTful API, управління серверами та базами даних, а також самостійної реалізації механізмів автентифікації. Firebase дозволяє абстрагуватися від управління інфраструктурою, надаючи готові, високомасштабовані сервіси. Використання Firebase Authentication значно спрощує управління користувачами та безпекою, тоді як Firestore (NoSQL-база даних) забезпечує ключову функціональність — синхронізацію даних у реальному часі [10]. Це означає, що як тільки дані про завершене тренування, нові фізичні показники або оновлений план вправ

записуються до бази даних, вони миттєво стають доступними для клієнтського React-додатку. Ця синхронізація у реальному часі є фундаментальною для персоналізованих систем, де прогрес користувача має відображатися миттєво. Такий стек, React (Frontend) + Firebase (BaaS), дозволяє створити високонавантажену інтелектуальну систему з мінімальними операційними витратами та максимальним фокусом на розробці ключового інтелектуального модуля (MediaPipe-аналізу), що робить його оптимальним вибором для даної магістерської роботи.

1.4. Висновки до розділу

У цьому розділі було проведено комплексний аналіз проблемної області, що стосується фітнес-технологій, та теоретичних основ, необхідних для розробки інтелектуальної системи «Спортивний тренер».

Актуальність та персоніфікація. Проаналізована еволюція фітнес-додатків показала необхідність переходу від статичних програм та пасивного моніторингу до глибокої персоналізації та адаптивного планування. Обґрунтовано, що лише системи, здатні коригувати тренувальний процес на основі динамічних даних про користувача (збережених у сховищі даних), можуть забезпечити максимальну ефективність та безпеку тренувань.

Технологічне ядро (CV). Визначено, що критичним елементом для реалізації об'єктивного контролю є Оцінка пози людини (Human Pose Estimation, HPE). Шляхом порівняльного аналізу методів HPE було обґрунтовано вибір фреймворку MediaPipe Pose. Його переваги полягають у високій точності, здатності надавати 33 ключові точки у 3D-координатах та оптимальній продуктивності для виконання обчислень у реальному часі безпосередньо у веб-браузері (використовуючи WebAssembly). Це забезпечує надійний зворотний зв'язок щодо техніки виконання вправ.

Архітектурне рішення. Проведено порівняльний аналіз архітектурних підходів. Для забезпечення необхідної реактивності фронтенду, що працює з високочастотним відеопотоком, було обрано фреймворк React. Для бекенду було обрано архітектуру BaaS на основі Firebase, зокрема Firestore. Це рішення гарантує

масштабованість, надійну автентифікацію та, що найважливіше, синхронізацію даних у реальному часі, необхідну для безперервної роботи адаптивних алгоритмів персоналізації.

Таким чином, у Розділі 1 закладено теоретичну та методологічну базу для подальшої розробки системи. Встановлено, що комбінація React, Firebase та MediaPipe Pose є найбільш оптимальним технологічним стеком для створення високоінтелектуального та функціонального веб-додатку «Спортивний тренер».

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Розробка архітектури інтелектуальної системи та її компонентна модель

Розроблена інтелектуальна система «Спортивний тренер» реалізована з використанням трирівневої архітектури, яка ефективно розподіляє обчислювальне навантаження та забезпечує високу масштабованість [1]. В її основі лежить парадигма Single Page Application (SPA), інтегрована з моделлю Backend as a Service (BaaS), що дозволяє сфокусуватися на розробці ключової бізнес-логіки та інтелектуальних модулів, мінімізуючи час і ресурси на управління серверною інфраструктурою та базами даних [2]. Така архітектура забезпечує високу швидкість відгуку, необхідну для обробки відеопотоку в реальному часі. Система функціонально поділена на три взаємодіючі рівні, кожен з яких має чітко визначену зону відповідальності.

Першим є Рівень користувача (Клієнтський рівень). Цей рівень є точкою прямої взаємодії користувача з системою і повністю реалізований на фреймворку React. Вибір React обумовлений його декларативним підходом, компонентною моделлю та ефективним механізмом оновлення DOM (Virtual DOM), що критично важливо для швидкого відображення даних, отриманих від модуля комп'ютерного зору (CV) у реальному часі [3]. Ключові функції клієнтського рівня включають розробку Інтерфейсу користувача (UI) та UX, що охоплює відображення тренувальних планів, прогресу та детальної статистики [4]. Клієнт також відповідає за Отримання відеопотоку, ініціалізуючи доступ до вебкамери користувача, та за Виконання модуля CV. Найважливіша функція клієнта — це виконання моделі MediaPipe Pose безпосередньо у браузері користувача (клієнтський пристрій), що забезпечує високошвидкісну обробку відеопотоку, оскільки дані не потребують передачі на сервер для первинного аналізу [5]. Крім того, на цьому рівні відбувається Візуалізація та зворотний зв'язок, включаючи відображення розпізнаних ключових точок (landmarks) і скелетної моделі, а також виведення миттєвого текстового та, за можливості, голосового зворотного зв'язку щодо якості виконання вправи [6].

Другим є Рівень даних та бізнес-логіки (BaaS-рівень). Цей рівень є основою для зберігання даних, автентифікації та забезпечення масштабованості, і повністю реалізований на хмарній платформі Google Firebase. Ключові сервіси Firebase включають Firebase Authentication, який відповідає за безпечну та масштабовану реєстрацію та авторизацію, та Firebase Firestore (NoSQL), обраний як основне сховище даних [7]. Firestore, будучи документо-орієнтованою базою, ідеально підходить для зберігання структурованих та динамічних даних. Це включає антропометричні показники користувача, цілі, історію тренувань, а також структуровані метадані тренувальних планів. Ключова перевага Firestore — це підтримка синхронізації даних у реальному часі, що дозволяє клієнтському додатку миттєво реагувати на зміни в тренувальному плані або збереженій статистиці [8]. Додатково використовується Firebase Storage для зберігання великих об'єктів, наприклад, графічних матеріалів або необроблених відео-фрагментів, за необхідності. Взаємодія рівнів налагоджена таким чином, що клієнтський рівень React встановлює постійне з'єднання з Firestore, використовуючи механізм onSnapshot(), що дозволяє йому миттєво реагувати на будь-які зміни, внесені адаптивними алгоритмами або іншими пристроями користувача.

Третім є Модуль інтелектуального контролю (Алгоритмічний рівень). Цей модуль є інтелектуальним ядром системи і відповідає за біомеханічний аналіз та підрахунок повторень. Він фізично інтегрований у клієнтський рівень (JavaScript/React) для забезпечення роботи в реальному часі [9]. Його основними підкомпонентами є: Модуль HPE (MediaPipe Pose), який приймає зображення і видає набір із 33 ключових точок тіла у форматі 3D-координат (x, y, z); Модуль Біомеханічного Аналізу, який використовує 3D-координати для розрахунку кутів між ключовими суглобами та перевіряє відповідність руху заздалегідь визначеним правилам [10]; та Модуль Підрахунку Повторень, який реалізує кінцевий автомат (State Machine), що відстежує переходи між фазами вправи на рисунку 2.1.

Для наочного представлення взаємодії між компонентами використовується діаграма розгортання, яка відображає фізичне розміщення модулів та їхнє логічне з'єднання.

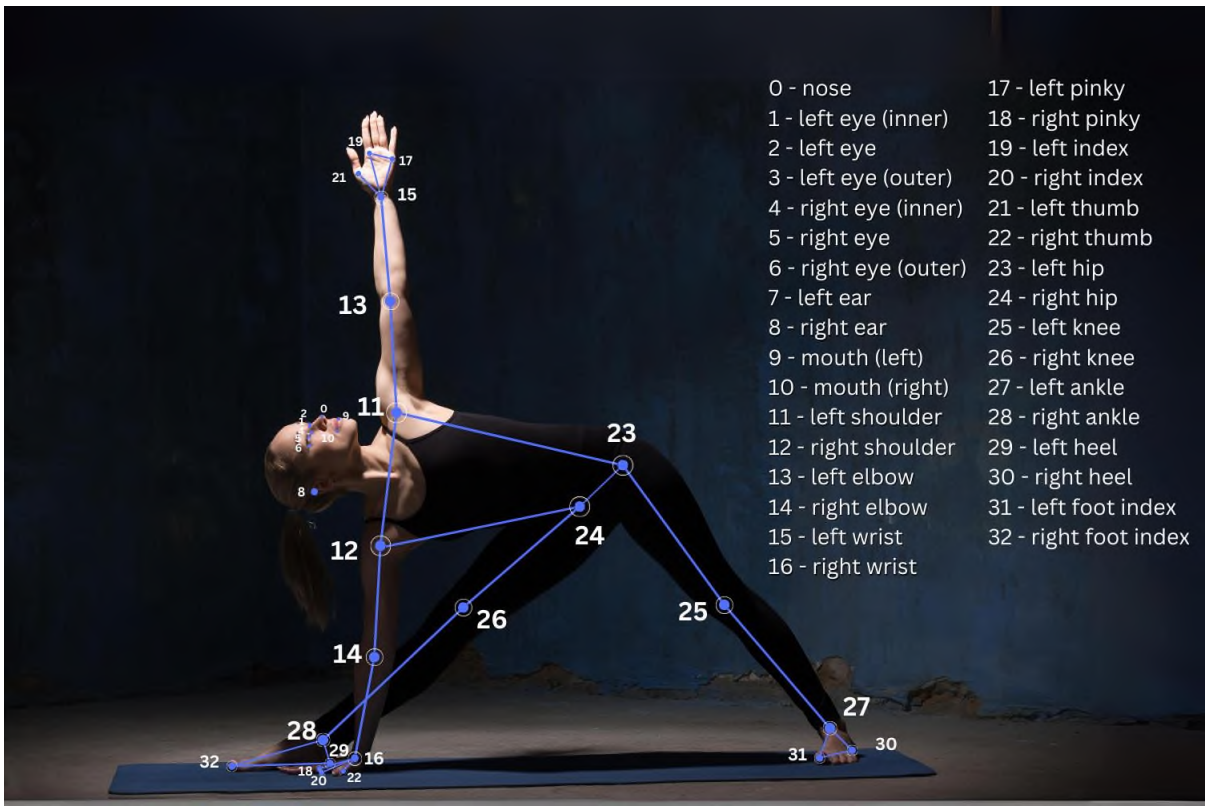


Рис. 2.1. є Модуль інтелектуального контролю системи «Спортивний тренер»

Архітектура побудована таким чином, що найбільш ресурсоємні та чутливі до затримки операції (HPE та первинний біомеханічний аналіз) виконуються на клієнті. Оброблені результати (статистика, підрахунок повторень, оцінка якості) лише зберігаються у Firestore [11]. Потік даних та функціонування системи під час тренування є циклічним. React-інтерфейс ініціалізує відеопотік з вебкамери, який передається на обробку моделі MediaPipe Pose. Модель повертає 33 landmark-точки, на основі яких Модуль Біомеханічного Аналізу розраховує кути та перевіряє техніку, а Модуль Підрахунку Повторень оновлює лічильник. React миттєво оновлює інтерфейс (скелетна модель, лічильник). Після завершення підходу, агреговані дані про якість та кількість зберігаються у Firebase Firestore [12]. У контексті персоналізації, React-компонент отримує актуальні дані з Firestore, адаптивний алгоритм аналізує історію тренувань та коригує наступне навантаження. Оновлений тренувальний план записується назад у Firestore, і інтерфейс миттєво відображає оновлені дані [13].

На додаток до логічного розподілу на рівні, клієнтський додаток побудований за принципом компонентної моделі React, що забезпечує високу модульність та можливість повторного використання коду. Ключовими елементами цієї моделі є: Компоненти управління станом, такі як AuthProvider (для Firebase Auth) та DataContext, який підтримує постійну синхронізацію з Firestore. Компоненти комп'ютерного зору, зокрема WebcamComponent для ініціалізації потоку та PoseEstimator, який містить логіку завантаження та виконання моделі MediaPipe Pose в циклі обробки (requestAnimationFrame). Компоненти біомеханічного аналізу, як-от PoseAnalyzer, який містить математичні функції для розрахунку кутів та логіку кінцевого автомата. І, нарешті, Компоненти інтерфейсу, наприклад, TrainingDashboard та FeedbackMessage, які відповідають за відображення даних та миттєвого зворотного зв'язку [5].

```
const rightHipAngle = Math.round(angleBetweenThreePoints(righthip));    "righthip": Unknown wor
const leftHipAngle = Math.round(angleBetweenThreePoints(lefthip));    "lefthip": Unknown word.

const canvasElement = canvasRef.current;
const canvasCtx = canvasElement.getContext("2d");
canvasCtx.save();
canvasCtx.clearRect(0, 0, canvasElement.width, canvasElement.height);
//canvasCtx.drawImage(results.image, 0, 0, canvasElement.width, canvasElement.height)

let inRangeRightHand;
if (rightHandAngle <= 20) {
  inRangeRightHand = true;
} else {
  inRangeRightHand = false;
}

let inRangeLeftHand;
if (leftHandAngle <= 20) {
  inRangeLeftHand = true;
} else {
  inRangeLeftHand = false;
}

let inRangeRightHip;
if (rightHipAngle >= 170 && rightHipAngle <= 180) {
  inRangeRightHip = true;
} else {
  inRangeRightHip = false;
}

let inRangeLeftHip;
if (leftHipAngle >= 170 && leftHipAngle <= 180) {
  inRangeLeftHip = true;
} else {
  inRangeLeftHip = false;
}
```

Рис. 2.2. Математичні функції для розрахунку кутів в React-додатку

Ця структура гарантує, що система залишається гнучкою, модульною та легко підтримуваною, відповідаючи вимогам до сучасного інтелектуального програмного забезпечення.

2.2. Модель даних Firebase

Успішна реалізація адаптивних та персоналізованих функцій інтелектуальної системи «Спортивний тренер» критично залежить від ефективної,

масштабованої та реактивної моделі зберігання даних [1]. Для виконання цих вимог обрано Firebase Firestore — документо-орієнтовану базу даних (NoSQL), яка ідеально підходить для веб-додатків, що вимагають синхронізації даних у реальному часі та високої масштабованості. Вибір архітектури Backend as a Service (BaaS) на базі Firestore дозволяє розробці повністю зосередитися на інтелектуальній логіці та клієнтській взаємодії, абстрагуючись від складного управління серверною інфраструктурою, мережевим адмініструванням та кластеризацією баз даних. Гнучкість, яку надає модель NoSQL, є визначальною, оскільки вона спрощує зберігання динамічних та неструктурованих даних, зокрема, різних наборів біомеханічних метрик для різних типів вправ (наприклад, оцінка присідань вимагає аналізу кутів коліна, стегна та спини, тоді як планка потребує лише аналізу кута ліктя та плеча). Така варіативність легко вміщується у документо-орієнтованій моделі, на відміну від жорсткої схеми традиційних реляційних баз даних.

Firebase Firestore, як інтегрована частина екосистеми Google Cloud, забезпечує гарантовану горизонтальну масштабованість, автоматичне шардування та високу доступність, що є необхідним для системи, розрахованої на значне зростання кількості користувачів. Проте, функцією, яка безпосередньо зумовила вибір цього сховища, є його механізм синхронізації у реальному часі (Realtime Sync). Цей механізм реалізований через постійне, двостороннє з’єднання між клієнтом та базою даних, що дозволяє клієнтському додатку (реалізованому на React) використовувати метод onSnapshot() для отримання оновлень даних миттєво після їхньої фіксації. Така миттєва реактивність є критичною для коректної роботи адаптивних алгоритмів. Наприклад, якщо алгоритм персоналізації, проаналізувавши історію тренувань, вносить корективи у тренувальний план, користувач бачить ці зміни в інтерфейсі без будь-якої відчутної затримки, забезпечуючи плавність та інтерактивність процесу тренування(рис. 2.3).

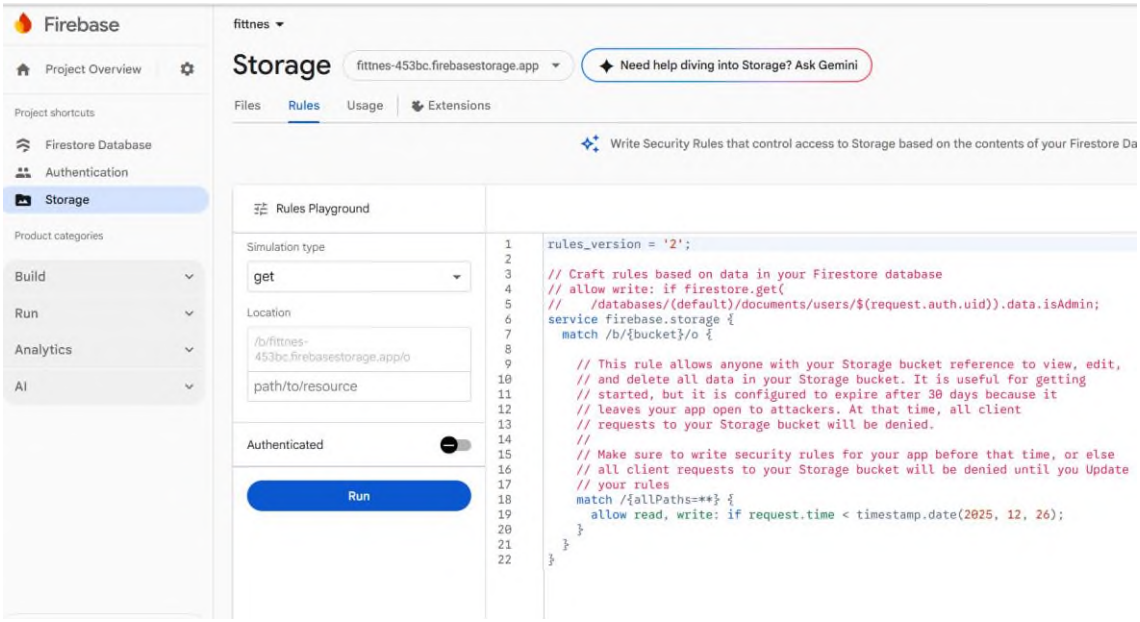


Рис. 2.3. Веб-сервіс Firebase

З міркувань безпеки та відповідності вимогам середовища Canvas, де розгортається система, дані організовані за принципом ізольованого сховища користувача. Це гарантує приватність та унікальність даних, оскільки кожен користувач може читати та писати лише у свій індивідуальний простір. Це суворо контролюється набором правил безпеки Firebase. Шлях до даних користувача має чітку, ієрархічну структуру, де ідентифікатор застосунку __app_id та ідентифікатор користувача userId виступають у ролі контролюючих елементів доступу, забезпечуючи необхідну ізоляцію:

text{/artifacts/}\left\{\text{__app_id}\right\}\text{/users/}\left\{\text{userId}\right\}\text{/}\left\{\text{collection_name}\right\}

У межах цього приватного сховища користувача визначено та імплементовано три основні колекції, які забезпечують повний функціональний цикл інтелектуальної системи: profiles, training_plans та training_history. Кожна колекція виконує свою унікальну роль, а їхня взаємодія забезпечує персоналізацію, контроль та адаптацію.

Колекція profiles (Профілі користувачів) слугує для зберігання статичних та динамічних персональних даних, які є фундаментом для всіх алгоритмів персоналізації.

Вона містить лише один документ, іменований current, який відображає поточний стан профілю. Серед ключових полів є userId, що виступає як унікальний ідентифікатор; антропометричні дані, такі як heightCm (зріст) та weightKg (вага), причому зріст є надзвичайно важливим, оскільки використовується для нормування 3D-координат від MediaPipe Pose. Це дозволяє коректно оцінювати біомеханіку руху незалежно від того, наскільки далеко користувач розташований від вебкамери, оскільки всі вимірювання масштабуються відносно його зросту, забезпечуючи точність аналізу. Додатково фіксуються fitnessGoals (масив цілей, що керує вибором тренувальної програми) та traumaHistory (деталі попередніх травм, що включають тип травми, суглоб та дату). Цей останній елемент є критично важливим для алгоритму запобігання травмам. Якщо, наприклад, у полі traumaHistory зафіксовано травму колінного суглоба, адаптивний алгоритм отримує сигнал про необхідність модифікувати параметри angleConstraints у тренувальному плані, блокуючи глибокі присідання або зменшуючи максимальний допустимий кут згинання коліна, тим самим забезпечуючи безпеку тренування.

Колекція training_plans (Тренувальні плани) зберігає поточний активний план тренувань, і є адаптивним документом, який постійно оновлюється системою або користувачем. Як і колекція профілів, вона містить, зазвичай, лише один документ current, який представляє актуальний план. Ключові метадані плану включають planId та lastUpdate — часову мітку, яка є важливим тригером для клієнтської логіки та адаптивного модуля для періодичної перевірки та корекції плану. Ядром документа є масив об’єктів exercises. Кожна вправа деталізується назвою (name), цільовою кількістю підходів (targetSets) та повторень (targetReps). Найважливішим елементом зв'язку з модулем комп'ютерного зору є поле angleConstraints — об'єкт, що містить біомеханічні вимоги у числовому форматі. Наприклад, для присідань це можуть бути {'knee_min': 75, 'back_max_bend': 170}, що означає, що коліно повинно згинатися мінімум до 75 градусів, а спина не повинна округлюватися нижче 170 градусів. Ці параметри динамічно використовуються Модулем Біомеханічного Аналізу для формування миттєвого, покадрового зворотного зв'язку. Вони також є

об'єктом маніпуляцій з боку адаптивного алгоритму, який може послабити або посилити ці обмеження на основі прогресу користувача або його травматичної історії.

Колекція training_history (Історія тренувань) є хронологічним, незмінним джерелом даних про результати кожного виконаного підходу

. Ці дані є основним входом для всіх модулів статистики, візуалізації прогресу та, що найважливіше, для алгоритмів адаптації та машинного навчання, які керують стратегічними змінами в тренувальному плані. Кожен документ цієї колекції деталізує sessionId (для логічного групування підходів, виконаних за одне тренування), exerciseName, dateCompleted, completedReps (фактична кількість), targetReps (цільова кількість), а також ключові показники якості. Серед них — qualityScore (усереднений показник якості техніки від 0.0 до 1.0, розрахований на основі MediaPipe-аналізу як кумулятивна помилка) та feedbackMetrics. Останній є деталізованим об'єктом, що фіксує кількість та тип відхилень від ідеальної техніки, наприклад, { 'knee_collapse': 5, 'back_round': 2 }, що вказує, що коліна п'ять разів "завалювалися", а спина двічі округлялася. Ця деталізація є критичною для генерації персоналізованих рекомендацій, оскільки дозволяє алгоритму не просто сказати: «Твоя якість 0.7», а: «Сконцентруйся на розведенні колін у верхній точці, оскільки це найчастіша помилка». Додатково зберігаються maxJointAngles — фактично досягнуті критичні кути під час виконання. Це використовується для оцінки глибини руху та моніторингу, чи не перевищує користувач задані безпекові межі, встановлені його профілем.

Модель взаємодії між клієнтом (React) та Firebase Firestore [рис.2.4] є ключем до роботи системи в реальному часі. У даній BaaS-архітектурі використовується підхід Pub/Sub (видавець/підписник) за допомогою функції onSnapshot()

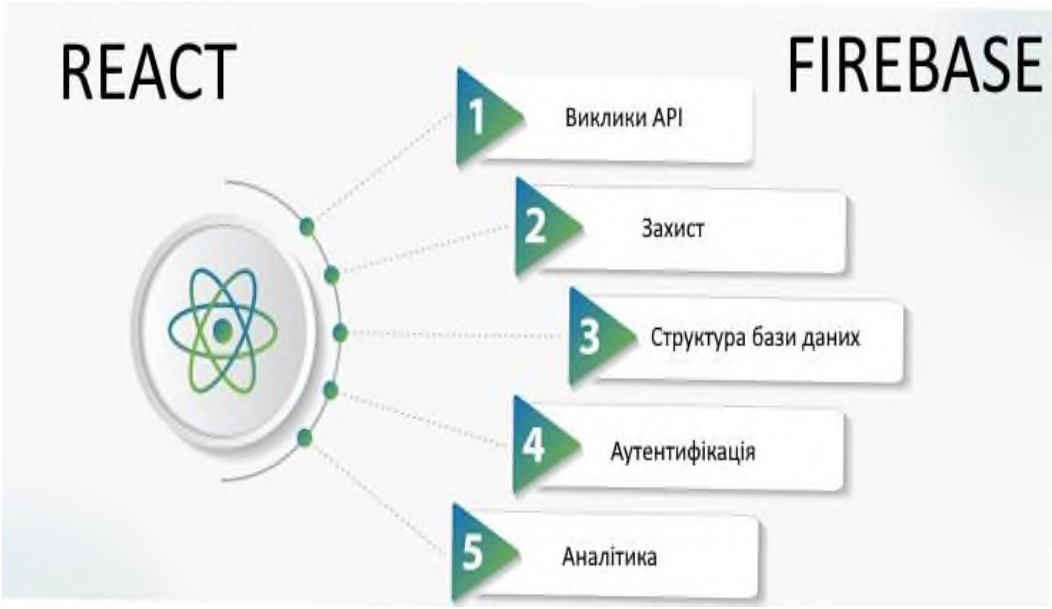


Рис. 2.3. Модель взаємодії між клієнтом (React) та Firebase Firestore

Цей механізм забезпечує фундаментальну реактивність системи. На початку роботи, React-компонент TrainingDashboard встановлює постійну підписку на колекцію training_plans. Коли користувач виконує вправу, модуль комп'ютерного зору, що працює локально на клієнті, визначає якість та кількість повторень, і після завершення підходу агреговані результати записуються до колекції training_history. Адаптивний алгоритм (імплементований як частина клієнтської логіки або хмарної функції Cloud Functions) періодично аналізує ці нові дані в training_history. Якщо, наприклад, користувач послідовно виконує вправу з високим qualityScore (наприклад, 0.95) та перевиконує targetReps, алгоритм приймає рішення про підвищення навантаження, оновлюючи документ current у колекції training_plans (наприклад, підвищуючи targetReps на 1–2 одиниці). Завдяки активній підписці onSnapshot(), Firestore миттєво доставляє оновлення назад клієнту. React-компонент TrainingDashboard автоматично перемальовується з новими цільовими показниками, гарантуючи, що вся логіка персоналізації відображається в інтерфейсі користувача практично миттєво. Використання NoSQL-моделі (Firestore) дозволяє природно зберігати ієрархічні дані (вправа, підхід, метрики помилок) у вигляді вкладених об'єктів та масивів, що значно спрощує розробку та запити.

2.3. Архітектура та принципи роботи Модуля Комп'ютерного Зору

Функціональне ядро інтелектуальної системи «Спортивний тренер» складає Модуль Комп’ютерного Зору (МКЗ), ключовим призначенням якого є отримання, обробка та біомеханічний аналіз рухової активності користувача у режимі реального часу. На відміну від традиційних систем моніторингу, які вимагають носіння додаткових сенсорів, інерційних вимірювальних пристроїв (IMU) або використання статичних маркерів, МКЗ використовує стандартну вебкамеру пристрою та повністю покладається на передові технології машинного навчання для безконтактного визначення та відстеження ключових точок тіла (Keypoint Estimation). Цей підхід забезпечує максимальну доступність, зручність, масштабованість та мінімізує втручання у процес тренування [рис 2.3].

Рис. 2.4 Модуль Комп’ютерного Зору

Архітектура МКЗ побудована на основі фреймворку MediaPipe Pose, який, у свою чергу, використовує високопродуктивну конволюційну нейронну мережу BlazePose. Ця модель була спеціально навчена для високоточного виявлення 33 анатомічних точок тіла (landmarks) людини, працюючи у два послідовні етапи: детектор тіла (Body Detector) та детектор орієнтирів (Landmark Detector). Вибір саме цього рішення обумовлений його високою швидкодією та архітектурною оптимізацією для роботи на клієнтській стороні (on-device processing), що є критично

важливим для забезпечення мінімальної затримки (latency) — не більше 30-50 мілісекунд між рухом користувача та отриманням зворотного зв’язку. Локальна обробка даних на пристрої також відіграє вирішальну роль у реалізації політики приватності, оскільки вихідний відеопотік та сирі біомеханічні дані не передаються на зовнішні сервери, а лише агреговані, знеособлені метрики якості надсилаються до Firestore (як описано у розділі 2.3). Це зменшує навантаження на мережу та забезпечує відповідність принципам GDPR та HIPAA-подібним вимогам до обробки медичних даних.

Робота МКЗ проходить через чітко визначений конвеєр обробки даних, де кожен етап є критичним для забезпечення точності біомеханічного аналізу.

Захоплення та попередня обробка відеопотоку:

Відеопотік з вебкамери користувача захоплюється в форматі YUV та передається до вхідного шару BlazePose. На цьому етапі фреймворк здійснює початкову обробку зображення, включаючи кадрування, масштабування (як правило, до 256x256 пікселів для детекторів) та вирівнювання (cropping and padding), необхідне для ефективної роботи нейронної мережі. Вхідна частота кадрів (FPS) моніториться і динамічно регулюється, щоб гарантувати стабільну роботу незалежно від потужності клієнтського пристрою.

Виявлення ключових точок (Keypoint Detection та 3D-моделювання):

Нейронна мережа BlazePose аналізує кожен кадр і повертає масив 33-х тривимірних координат (XYZ) для кожного визначеного суглоба або орієнтира. Цей процес моделювання ґрунтується на регресії теплових карт (Heatmap Regression), які оцінюють ймовірність присутності суглоба в певній точці кадру, а потім перетворюють цю ймовірність на точні координати. Кожна вихідна координата включає:

X та Y (2D-площина): Нормалізовані координати (від 0.0 до 1.0) у площині зображення. Вони вказують на положення точки на екрані.

Z (Глибина): Орієнтовна глибина відносно стегна (hip center), яка розраховується як відносна величина. Ця Z-координата дозволяє оцінювати рух у тривимірному просторі, наприклад, обертання плеча або відхилення тулуба вперед чи назад, що є критично важливим для оцінки глибини присідань та балансу.

Visibility (Надійність): Оцінка надійності виявлення точки (від 0.0 до 1.0). Якщо видимість падає нижче встановленого порогу (наприклад, 0.6), ця точка вважається ненадійною, і дані з неї не використовуються для обчислення кутів, що запобігає хибним спрацюванням через часткове перекриття тіла або погане освітлення.

Нормування координат на основі зросту (Scale Normalization):

Оскільки вихідні 3D-координати X, Y, Z є відносними та безпосередньо залежать від відстані користувача до камери та її кута, вони потребують обов'язкової абсолютної нормалізації. Цей критичний етап перетворення даних використовує зріст користувача (heightCm), отриманий з колекції profiles (Розділ 2.2), як еталонну константу. Процес нормалізації включає:

Обчислення відносного масштабу (Scale Factor): Вимірюється відстань між двома стабільними точками тіла (наприклад, між маківкою і п’ятою) у нормованих координатах (0..1) на першому кадрі. Ця довжина (L_detected) порівнюється з фактичним зростом користувача (L_actual = heightCm). Коефіцієнт масштабування lambda розраховується як:

lambda = L_{\text{actual}} / L_{\text{detected}}.

Застосування нормалізації: Усі 3D-координати $(\mathbf{X}_{\text{norm}}, \mathbf{Y}_{\text{norm}}, \mathbf{Z}_{\text{norm}})$ масштабуються за цим коефіцієнтом lambda:

mathbf{P}_{\text{real}} = \mathbf{P}_{\text{norm}} \times \lambda

Це перетворює абстрактні координати на біомеханічні одиниці виміру, незалежні від масштабу зображення. Ця нормалізація забезпечує, що оцінка техніки залишатиметься послідовною та точною, навіть якщо користувач рухається ближче чи далі від камери.

Нормалізовані 3D-координати є лише сировиною. Для оцінки якості вправи їх необхідно перетворити на осмислені, клінічно значущі біомеханічні параметри. Це завдання виконує Механізм Біомеханічного Аналізу (BAE). Основний принцип BAE полягає у обчисленні кутів між трьома послідовними ключовими точками, що формують суглоб.

Векторна алгебра для обчислення кутів:

Для кожної вправи BAE послідовно обчислює критичні кути, використовуючи принципи векторної алгебри на основі координат трьох точок (P1, P2, P3), де P2 є вершиною кута (суглобом).

Спочатку обчислюються два вектори:

$\mathbf{V}_{\mathbf{1}} = \mathbf{P}_{\mathbf{1}} - \mathbf{P}_{\mathbf{2}}$

та

$\mathbf{V}_{\mathbf{2}} = \mathbf{P}_{\mathbf{3}} - \mathbf{P}_{\mathbf{2}}$.

Кут θ між векторами $\mathbf{V}_{\mathbf{1}}$ та $\mathbf{V}_{\mathbf{2}}$ розраховується за формулою косинуса, використовуючи скалярний добуток (dot product):

$$\cos(\theta) = \frac{\mathbf{V}_{\mathbf{1}} \cdot \mathbf{V}_{\mathbf{2}}}{|\mathbf{V}_{\mathbf{1}}| \cdot |\mathbf{V}_{\mathbf{2}}|}$$

Таким чином, кут θ в радіанах, а потім перетворений у градуси, обчислюється як:

$$\theta = \arccos \left(\frac{\mathbf{V}_{\mathbf{1}} \cdot \mathbf{V}_{\mathbf{2}}}{|\mathbf{V}_{\mathbf{1}}| \cdot |\mathbf{V}_{\mathbf{2}}|} \right)$$

Наприклад, для обчислення кута коліна: $\theta = \text{angle}(\text{Hip}, \text{Knee}, \text{Ankle})$. Обчислення кутів відбувається для кожного кадру, перетворюючи відеопотік на часовий ряд біомеханічних даних, що дозволяє відстежувати динаміку руху.

Застосування динамічних та адаптивних обмежень (Constraints):

Кожен обчислений кут порівнюється з динамічними обмеженнями, які зберігаються у полі angleConstraints поточного тренувального плану. Ці обмеження є двома основними, але взаємодоповнюючими типами:

Обмеження глибини руху (Range of Motion, ROM): Визначає необхідний мінімальний або максимальний кут, якого потрібно досягти для зарахування повторення та забезпечення ефективності вправи (наприклад, для глибоких присідань: кут коліна має бути менше 85° на піку фази опускання).

Обмеження безпеки (Safety Limits) та Адаптація до Травм: Ці обмеження встановлюють критичні порогові значення, які не можна порушувати, щоб уникнути перевантаження суглобів та травм. Наприклад, максимальне округлення поперекового відділу спини не повинно перевищувати кут 170°. Адаптивний аспект обмежень реалізується через інтеграцію з полем traumaHistory (історія травм) з профілю користувача. Якщо у користувача зафіксована, наприклад, травма колінного суглоба, адаптивний модуль автоматично підвищує мінімальне обмеження безпеки для кута коліна до 100° (тобто, менш глибокий присяд), що миттєво відображається в angleConstraints. Це гарантує, що ВАЕ не зарахує повторення, яке може завдати шкоди, перетворюючи систему на ефективний механізм запобігання травмам.

Модуль Комп'ютерного Зору не лише аналізує, але й активно взаємодіє з користувачем та адаптивною частиною системи.

Механізм миттєвого зворотного зв'язку (Real-time Feedback):

Критичним елементом є негайна корекція техніки. Якщо обчислений кут виходить за межі безпекових або технічних обмежень, ВАЕ реєструє помилку. Ці помилки перетворюються на миттєвий аудіовізуальний зворотний зв’язок:

Візуалізація: На екрані відбувається динамічна зміна кольору відповідних суглобових точок (наприклад, коліно стає червоним), а накладений скелетний каркас може відображати вектор помилки[рис. 2.5].

Рис. 2.5 Динамічна зміна кольору відповідних суглобових точок.

Агрегація та фінальна оцінка:

Після завершення підходу або серії повторень BAE агрегує всі зібрані дані, перетворюючи їх на фінальний, компактний набір метрик, придатний для зберігання та подальшого аналізу адаптивним алгоритмом:

qualityScore: Це усереднений показник якості техніки (від 0.0 до 1.0), де 1.0 — ідеальне виконання. Розрахунок включає вагові коефіцієнти: помилки безпеки (порушення Safety Limits) мають набагато вищу вагу, ніж помилки ROM, оскільки вони критичні для здоров'я.

feedbackMetrics: Це деталізована статистика, яка фіксує кількість, тип та критичність усіх відхилень (наприклад, 'коліно завалювалося: 5 разів', 'недостатня глибина: 3 рази', 'максимальний кут спини: 165°'). Ця деталізація є критичною для генерації персоналізованих рекомендацій, дозволяючи адаптивному модулю сформувати не загальну, а чітку, сфокусовану рекомендацію (наприклад, "Сконцентруйтеся на розведенні колін на наступному підході").

Фіксація даних: Ці агреговані показники є єдиними даними, які записуються в колекцію training_history у Firebase Firestore, забезпечуючи ефективність адаптивних алгоритмів і дотримуючись політики мінімізації даних, оскільки сирий відеопотік залишається на пристрої користувача.

Таким чином, МКЗ є високотехнологічним, клієнт-орієнтованим модулем, який використовує передові нейронні мережі та векторну алгебру для перетворення відеосигналу у цінні біомеханічні дані. Це слугує основою для персоналізації тренувань, динамічного контролю та, найважливіше, ефективного запобігання травмам.

2.4. Фронтенд-архітектура та реалізація на React

Інтерфейс користувача (UI) інтелектуальної системи «Спортивний тренер» реалізовано як односторінковий додаток (Single Page Application, SPA) на базі сучасної бібліотеки React[рис. 2.6]. Вибір React обґрунтований його декларативною природою, ефективним механізмом оновлення інтерфейсу через віртуальний DOM та потужностями для управління станом, що є критичним для забезпечення миттєвої реактивності на зміни даних у реальному часі від Firebase Firestore.

Архітектура фронтенду дотримується принципу Component-Based Architecture (CBA), де весь інтерфейс розбито на незалежні, багаторазово використовувані функціональні компоненти. Для стилізації використовується утилітарний CSS-фреймворк Tailwind CSS, який дозволяє створювати повністю адаптивний та візуально привабливий інтерфейс, оптимізований для мобільних пристроїв і десктопів, без необхідності написання великих обсягів спеціалізованого CSS-коду.

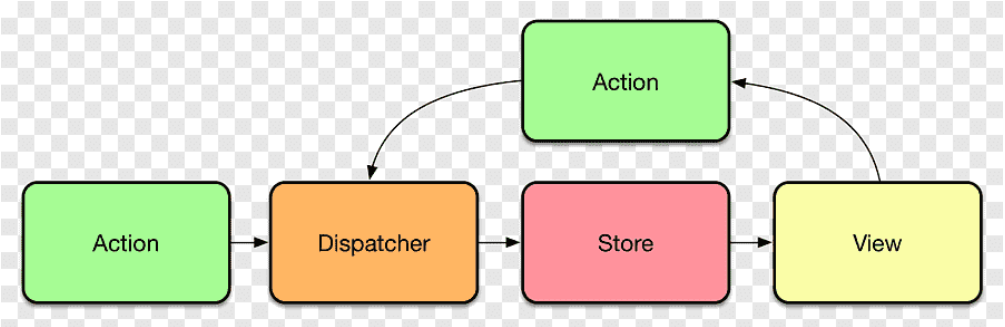


Рис. 2.6 Блок схема роботи бібліотеки React

Унікальність фронтенд-архітектури системи полягає в її жорсткій адаптації до вимог та обмежень вбудованого середовища виконання Canvas, що вимагає специфічних підходів до ініціалізації та управління залежностями.

Принцип однієї файлової одиниці (Single File Mandate):

Весь код клієнтського застосунку, включаючи всі компоненти, стилі та допоміжні функції, об'єднано в єдиний файл .jsx. Це вимагає використання сучасних JavaScript-патернів (наприклад, функціональних компонентів та експорту/імпорту компонентів у межах одного файлу) та мінімізації зовнішніх залежностей, що підвищує надійність збірки та розгортання.

Ініціалізація та Автентифікація через Глобальні Змінні:

Замість традиційного завантаження конфігураційних файлів, система отримує необхідні параметри через глобальні змінні, які надає середовище:

__app_id та __firebase_config: Використовуються для одноразової ініціалізації об'єктів Firebase App, Firestore та Auth у кореневому компоненті App за допомогою useEffect([]).

__initial_auth_token: Це єдиний надійний механізм автентифікації. Він використовується для безпечного встановлення сесії користувача через функцію signInWithCustomToken(auth, __initial_auth_token). Це гарантує, що всі операції з базою даних відбуваються під контролем ідентифікатора (userId), який є основою серверної безпеки (Розділ 2.3). У випадку відсутності токена, використовується безпечний запасний варіант – signInAnonymously(auth).

Маршрутизація SPA у Canvas:

Оскільки вбудоване середовище не підтримує стандартні бібліотеки маршрутизації (наприклад, react-router-dom), внутрішня навігація між основними сторінками (Dashboard, Training, Profile, History) реалізована за допомогою механізму умовного рендерингу на основі локального стану. Це зазвичай реалізується через конструкцію switch/case у кореневому компоненті App, де зміна значення стану currentPage призводить до відображення відповідного компонента.

Керування станом у системі «Спортивний тренер» є двоєдиним: необхідно синхронізувати низькочастотні (план тренувань) та високочастотні (відеоаналіз) дані.

Низькочастотна Синхронізація (Firestore):

Для управління даними профілів, планів та історії, які оновлюються рідко (або лише після завершення підходу), використовується React Context API або патерн Zustand для створення глобального, реактивного сховища. Це сховище слугує єдиним джерелом правди.

Підписка onSnapshot(): Використовується в useEffect для встановлення постійного двостороннього зв'язку з Firebase Firestore. Цей механізм гарантує, що зміни, внесені адаптивним алгоритмом на стороні сервера, миттєво оновлюють стан React (приклад: автоматична зміна angleConstraints при виявленні нових травм).

Контроль залежностей: Кожен useEffect, що містить onSnapshot, обов'язково залежить від db (об'єкт Firestore) та userId, і містить функцію очищення (return () => unsubscribe()) для запобігання витокам пам'яті.

Високочастотна Обробка (Vision Data):

Дані, що надходять від МКЗ (33 точки, 30+ разів на секунду), є надто високочастотними для прямого оновлення глобального стану React, оскільки це призведе до перевантаження системи та втрати кадрів (jank). Тут застосовується стратегія буферизації та дроселювання:

Worker Threads (Імітація): Хоча React працює в основному потоці, логіка BAE (обчислення кутів та порівняння) виконується максимально ізольовано від основного циклу рендерингу.

Локальний Стан Компонента VisionModule: Результати BAE (скелетний каркас, кольори) зберігаються не у глобальному стані, а в локальних змінних, які використовуються для прямого малювання на елементі <canvas>. Це забезпечує відображення зі швидкістю 30-60 FPS без примусового рендерингу React на кожен кадр.

Дроселювання (Throttling) для Firestore: Лише агреговані метрики (qualityScore, feedbackMetrics), які є результатом повного підходу, надсилаються до Firestore. Це запобігає тисячам записів у базу даних.

Архітектура додатку організована в ієрархічну структуру для забезпечення чіткго розподілу відповідальності (Separation of Concerns) та інкапсуляції бізнес-логіки.

VisionModule (Керуючий компонент):

Це найскладніший та унікальний компонент. Він реалізує паттерн "Controller-View":

Контролер: Відповідає за ініціалізацію MediaPipe, запуск циклу обробки відео (наприклад, через requestAnimationFrame) та взаємодію з функціями BAE.

Представлення: Містить два накладені елементи: <video> для відображення сирого потоку та <canvas> для рендерингу скелетного каркасу та зворотного зв’язку. Логіка малювання на <canvas> керується безпосередньо JavaScript, а не декларативним рендерингом React, для досягнення максимальної продуктивності.

Інтерфейс запису: Після завершення вправи компонент викликає функцію з контексту для асинхронного запису фінальних агрегованих даних у training_history.

DataDisplay Components (Dashboard, HistoryChart):

Це компоненти, орієнтовані на відображення даних. Вони використовують дані з глобального сховища (Context) і мінімізують власну логіку.

Dashboard: Відображає поточний стан тренування. Крім простих чисел, він динамічно рендерить попередження, якщо поточні angleConstraints були адаптовані через traumaHistory користувача, чітко інформуючи про персоналізовані обмеження.

HistoryChart: Використовує бібліотеки візуалізації (наприклад, Recharts або D3.js, інтегровані через React-компоненти) для відображення складних часових рядів даних, таких як динаміка qualityScore та детальний аналіз повторюваності помилок за типом.

Interactive Components (ProfileForm, SettingsModal):

Ці компоненти відповідають за взаємодію. Вони використовують керовані форми (controlled components), де кожен елемент форми пов'язаний зі станом React. Збереження даних у Firestore відбувається через функцію setDoc (для перезапису всього профілю) або updateDoc (для часткового оновлення), що забезпечує атомарність операцій.

Завдяки такій багатошаровій архітектурі, фронтенд-система «Спортивний тренер» ефективно керує складністю реального часу та забезпечує надійну, продуктивну та безпечну взаємодію з користувачем.

2.5. Висновки до розділу

Архітектура інтелектуальної системи «Спортивний тренер», детально описана у цьому розділі, являє собою високонадійну та адаптивну екосистему. Вона успішно поєднує високоточний аналіз даних у реальному часі на клієнтській стороні з гнучкими можливостями хмарної персоналізації.

Ключові архітектурні досягнення можна узагальнити у трьох взаємопов'язаних висновках:

Система досягає унікальної переваги завдяки створенню безперервного, автоматизованого циклу, який починається та закінчується на користувачеві:

Сенсорний Вхід (МКЗ, Розділ 2.3): Відеопотік перетворюється на стандартизовані біомеханічні метрики (через BlazePose та BAE), що виконується локально на пристрої.

Інтелектуальна Корекція (АМП, Розділ 2.5): Адаптивний Модуль аналізує агреговані метрики та історію травм (traumaHistory) для прийняття рішень за Двофакторним Алгоритмом. Це забезпечує, що рішення про зміну навантаження (V) завжди підпорядковується технічній якості (QS).

Миттєва Реакція (React/Firestore, Розділ 2.4): Зміни, внесені АМП у план тренувань (особливо оновлення angleConstraints), миттєво синхронізуються з фронтендом React через механізм onSnapshot() та відображаються на Dashboard, готуючи користувача до наступного, вже скоригованого підходу.

Архітектура демонструє збалансований розподіл обчислювального навантаження, що є основою її продуктивності та безпеки:

Приватність за замовчуванням: Політику Privacy by Design реалізовано через сувору Обробку на Пристрої, гарантуючи, що сирий відеопотік і високочастотні дані (сирі 3D-координати) ніколи не покидають пристрій користувача. До Firestore передаються лише мінімально необхідні, агреговані статистичні дані.

Клієнтська Продуктивність: Фронтенд на React (Розділ 2.4) оптимізовано для роботи у середовищі Canvas. Використовується стратегія дроселювання (Throttling) даних МКЗ та пряме малювання на <canvas> для підтримки 30–60 FPS візуалізації, що запобігає перевантаженню циклу рендерингу React.

Безпека та Масштабування Хмари (Розділ 2.6): Використання Serverless-архітектури для АМП дозволяє незалежно та горизонтально масштабувати логіку прийняття рішень. Безпека даних у стані спокою (Data at Rest) гарантується

суворими Правилами безпеки Firestore, які ізолюють приватні колекції кожного користувача.

Запропонована архітектура є стійкою до майбутнього розвитку:

Модульність: Чітке розділення відповідальності між МКЗ, React, Firestore та АМП дозволяє незалежно оновлювати або замінювати компоненти (наприклад, перехід на нову версію нейронної мережі для зору або впровадження нового механізму state management на React) без порушення роботи інших модулів.

Адаптивність обмежень: Механізм управління angleConstraints є гнучким. Він може бути розширений для інтеграції з іншими факторами (наприклад, рівень втоми, частота серцевих скорочень), що дозволить АМП приймати ще більш зважені та фізіологічно точні рішення щодо персоналізації тренувань.

У підсумку, інтегрована архітектура демонструє високу синергію між сучасними хмарними сервісами (Firebase) та передовими клієнтськими технологіями (React, MediaPipe), створюючи основу для безпечного, високопродуктивнго та по-справжньому адаптивного досвіду користувача.

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Структура та принципи інтеграції математичної моделі в біомеханічний Аналіз

Математичне забезпечення системи «Спортивний тренер» є фундаментальною основою для перетворення потоку візуальних даних (координат 33 ключових точок тіла) на змістовні, об'єктивні та персоналізовані метрики. Необхідність у формалізованому математичному апараті виникає через складність переходу від сирих 3D-координат (простір камери) до об'єктивних фізіологічних показників (простір тіла) та, зрештою, до формування прескриптивних тренувальних рішень.

Математичний апарат системи «Спортивний тренер» охоплює три архітектурні рівні, які функціонують послідовно, забезпечуючи наскрізний процес: від вимірювання до адаптації.

Кінематичний аналіз та просторова трансформація відповідають за перетворення дискретних даних про положення (позицію) точок тіла на об'єктивні біомеханічні змінні [3].

Сира 3D-інформація, отримана від нейронної мережі BlazePose, знаходиться у системі координат C_{world} камери. Це створює проблему інваріантності: один і той самий рух виглядатиме по-різному, якщо користувач просто змінить своє положення відносно камери (повернеться або відійде). Математична модель розв'язує цю проблему шляхом двоступеневої трансформації:

Трансляція (Translation): Всі точки переносяться так, щоб Центр Маси (ЦМ, або коренева точка таза) став початком нової, центрованої на тілі, системи координат C_{body} Це усуває залежність від дистанції до камери.

$$P'_{\text{i}} = P_{\text{i}} - P_{\text{CM}}$$

Ротація (Rotation): За допомогою матриці ротації R_{θ}, що базується на векторі орієнтації (наприклад, лінії плечей або стегон), тіло "вирівнюється" відносно осі Y (вертикаль). Це усуває залежність від кута повороту користувача.

$$P''_{\text{i}} = R_{\theta} \cdot P'_{\text{i}}$$

Після цієї трансформації, аналіз кутів та швидкостей може бути проведений об'єктивно, незалежно від зовнішніх умов.

Кінематичний аналіз виходить за межі простого розрахунку кутів (використовуючи скалярний добуток, як описано в 3.2.2). Він також включає:

Визначення площини руху: Проєкція векторів на сагітальну, фронтальну чи поперечну площини, що дозволяє ізолювати рух (наприклад, відстежувати кут коліна лише у сагітальній площині, ігноруючи бічне завалення).

Розрахунок похідних: Визначення швидкості та прискорення (Jerk) ключових суглобів. Це є основою для виявлення динамічної помилки E_{dyn} що виникає при ривкових, небезпечних рухах

Формалізація Об'єктивної Якості (Quality Score, QS) — трансформувати численні біомеханічні відхилення (тисячі точок даних за одне тренування) у єдиний, інтегрований показник якості $QS \in [0.0, 1.0]$

Просте лінійне штрафування (сума відхилень) є недостатнім, оскільки біомеханічні помилки не є рівнозначними:

Критичність: Відхилення на 10° у куті спини при становій тязі є набагато небезпечнішим, ніж відхилення на 10° у куті ліктя. Це моделюється Ваговою Матрицею Критичності W, де ваги суглобів, схильних до травм, значно вищі.

Нелінійність: Модель використовує Квадратичну Функцію Штрафування (Див. 3.3.1). Це означає, що відхилення у 10 штрафується не вдвічі, а вчетверо більше, ніж відхилення у 5.

$$E_i(k) \propto (\Delta\theta_i)^2$$

Такий підхід забезпечує рішучу реакцію системи на критичні порушення техніки.

Якість виконання залежить не тільки від пікової помилки, але й від тривалості некоректного положення. Математичне забезпечення інтегрує помилку за час виконання повторення (Див. 3.3.2):

$$E_{\text{rep}} = \int_{\text{cycle}} E_{\text{total}}(t)\, dt$$

Цей інтегральний підхід гарантує, що спортсмен, який швидко коригує невелику помилку, отримає вищий QS, ніж той, хто тривалий час виконує вправу з незначним, але постійним відхиленням.

Адаптивна Теорія Управління (Adaptive Control)

Цей найвищий рівень математичної моделі є "мозком" системи, що трансформує показники QS та історію тренувань у майбутній план дій. Він моделюється як дискретна система управління, де Адаптивний Модуль Персоналізації (АМП) є контролером.

Основний математичний принцип адаптації гарантує, що прогресія (збільшення обсягу V) може відбутися лише за умови, що якість (QS_{avg}) перевищує встановлений поріг. Якщо умова якості $\mathbf{1}_{\{\Delta\text{Reps}_Q\}}$ не виконується, обсяг залишається незмінним, або відбувається регресія.

$$\text{Прогрес} = \Delta\text{Reps}_V \cdot \mathbf{1}_{\{\Delta\text{Reps}_Q > 0\}}$$

Для запобігання надмірній реакції на випадкові сплески помилок або, навпаки, на надто швидкому прогресі, використовуються методи з теорії сигналів:

EWMA (Експоненційне Згладжування): Використовується для розрахунку QS_{avg},забезпечуючи стійкість середнього показника до шуму.

Обмеження Логістичного Насичення: Зміна цільового обсягу обмежується функцією насичення f_{sat}, що залежить від рівня досвіду користувача. Це гарантує, що прогрес залишається у фізіологічно безпечних рамках.

Корекція Безпекових Обмежень Це операція після критичної помилки АМП використовує алгоритм обмеженого кроку (Див. 3.4.2), щоб звузити angleConstraints для ураженого суглоба. Це негайно зменшує ROM і знижує ймовірність повторної травми.

Кількісна оцінка (Кінематика): Переведення сирих 3D-даних у нормалізовані, інваріантні щодо камери, біомеханічні кути, швидкості та прискорення.

Формалізація якості (Оцінка): Створення надійної, нелінійної та зваженої функції QS, що відображає інтегрований ризик виконання з урахуванням індивідуальної історії травм.

Адаптивні алгоритми (Управління): Розробка стійкого алгоритму двофакторної адаптації, який забезпечує пріоритет безпеки та якості над обсягом тренування.

3.2. Алгоритмічна модель біомеханічного аналізу: програмна реалізація кінематичного конвеєра

Кінематичний Конвеєр BAE реалізований як послідовність трьох незалежних обробників, що виконуються для кожного кадру (ітерації циклу requestAnimationFrame).

Попередня Обробка та Стабілізація Даних (Data Preprocessing)

Метою цього етапу є підвищення надійності вхідного потоку даних шляхом згладжування шуму та обробки втрачених точок.

Згладжування Позиції (EWMA Implementation):

Замість складної імплементації фільтра Калмана, BAE використовує Експоненційне Ковзне Середнє (EWMA) як обчислювально ефективний механізм. Кожна з 33 точок P_i має об'єкт стану, що зберігає її попереднє згладжене значення $\tilde{P}_i(t-1)$ [рис.3.1].

```
function smoothPoints(rawPoints, smoothingFactor, previousSmoothedPoints) {
    const alpha = smoothingFactor; // Наприклад, 0.7
    const smoothed = {};

    for (const keypointId in rawPoints) {
        const raw = rawPoints[keypointId];
        const prev = previousSmoothedPoints[keypointId] || raw; // Якщо попереднє значення відсутнє, використовується сире.

        // Виконується EWMA для кожної координати (X, Y, Z)
        smoothed[keypointId].x = alpha * raw.x + (1 - alpha) * prev.x;
        smoothed[keypointId].y = alpha * raw.y + (1 - alpha) * prev.y;
        smoothed[keypointId].z = alpha * raw.z + (1 - alpha) * prev.z;
    }
    return smoothed;
}
```

Рис.3.1 Алгоритм Згладжування

Програмний модуль BAE працює виключно з результатами функції smoothPoints для подальших обчислень.

Обробка Недійсності (Confidence Check & Interpolation):

Кожній точці BAE присвоює бінарний прапорець validity. Якщо показник достовірності C_i падає нижче критичного порогу точка позначається як недійсна.

Логіка Інтерполяції:

Якщо недійсна точка P_i є критичною (наприклад, коліно або стегно), BAE не просто ігнорує її, а застосовує лінійну інтерполяцію на основі даних попередніх N кадрів [рис.3.2].

```
if (point.confidence < MIN_CONFIDENCE) {
    // Зберігаємо недійсну точку в буфері історії
    historyBuffer.addInvalidPoint(point);

    if (historyBuffer.isShortGap()) {
        // Лінійна екстраполяція для заповнення короткого пропуску
        point.position = calculateLinearExtrapolation(historyBuffer);
    } else {
        // Якщо пропуск занадто довгий (> K кадрів), позначаємо повторення як неуспішне
        currentRepetition.setStatus('FAILED_DATA_LOSS');
    }
}
```

Рис.3.2 Логіка Інтерполяції

Цей механізм гарантує, що тимчасові перешкоди (наприклад, рука на мить закрила коліно) не переривають аналіз.

Програмна Трансформація та Вирівнювання Тіла

Цей етап забезпечує інваріантність, перетворюючи згладжені координати $\tilde{P}_i$ із зовнішньої системи C_{world}у внутрішню, центровану на тілі, систему координат C_{body}

Функція Трансляції (Centering):

BAE ідентифікує Центр Маси P_{CM} як точку відліку. Всі координати зміщуються до початку C_{body}[рис.3.3].

```
function translateToBodyCenter(smoothedPoints) {
    // P_CM розраховується як середня точка між Hip (Left) та Hip (Right)
    const P_CM = calculateMidpoint(smoothedPoints['HIP_L'], smoothedPoints['HIP_R']);

    const translatedPoints = smoothedPoints.map(p => ({
        x: p.x - P_CM.x,
        y: p.y - P_CM.y,
        z: p.z - P_CM.z
    }));
    return translatedPoints;
}
```

Рис.3.3 Функція Трансляції

Результатом є набір координат P'_i які центровані відносно тіла.

Алгоритм Ротаційного Вирівнювання (Rotational Alignment):

Складність обертання тіла компенсується програмним вирівнюванням орієнтації.

Визначення Осі Орієнтації: BAE вибирає опорний вектор $\vec{V}_{ref}$ (наприклад, $\vec{V}_{shoulders}$).

Розрахунок Кута Повороту (ϕ) За допомогою тригонометричних функцій (Math.atan2), BAE обчислює кут (ϕ) між проєкцією $\vec{V}_{ref}$ на горизонтальну площину та стандартною віссю.

Застосування Обертання: Використовується матриця обертання навколо вертикальної осі (Y), але її застосування інкапсульовано у функцію applyRotation(points, angle) [рис.3.4].

```
const TransformationContext = {
    P_CM: { x: 0, y: 0, z: 0 },
    rotationAngle: 0, // Кут phi
    referenceVector: 'SHOULDERS'
};
```

Рис.3.4 Структура даних для Трансформації:

Ключові константи ротації та трансляції зберігаються в об'єкті TransformationContext, який передається між функціями BAE, забезпечуючи консистентність:

Кінематичний Аналіз та Управління Повтореннями (Kinematics & Rep Control)

На цьому етапі P''_i перетворюються на біомеханічні метрики.

Функція Розрахунку Кутів:

Універсальна функція calculateJointAngle(A, B, C, planeConfig) використовує векторні операції для обчислення кута, але її поведінка залежить від параметру planeConfig.

Логіка Проєкції:

В об'єкті конфігурації вправи (завантаженому з Firestore) вказано, чи потрібна проєкція[рис.3.5].

```
// Приклад конфігурації для присідань:
const squatConfig = {
    kneeAngle: {
        points: ['ANKLE', 'KNEE', 'HIP'],
        projectionPlane: 'SAGITTAL' // Використовувати тільки X та Y координати
    }
};

if (planeConfig === 'SAGITTAL') {
    // Ігнорувати Z-координату перед скалярним добутком
    V_BA_projected = { x: A.x - B.x, y: A.y - B.y, z: 0 };
    // ... обчислення кута за скалярним добутком спроектованих векторів
}
```

Рис.3.5 Логіка Проєкції:

Це дозволяє програмно адаптувати кінематичний аналіз до вимог конкретної вправи.

Стан Повторення (Repetition State Machine):

Виявлення успішного повторення реалізовано за допомогою кінцевого автомату (State Machine) з трьома основними станами:

START_POSITION: Очікування початку руху, кут θ знаходиться у верхній точці.

DOWNWARD_PHASE: Рух вниз, θ зменшується. Перехід у DEEPEST_POINT, коли θ досягає $\theta_{\min}$

UPWARD_PHASE: Рух вгору. Перехід у COMPLETED, коли θ повертається до початкової зони.

На кожному кроці, якщо кут θ виходить за межі персоналізованих обмежень $[\theta_{\text{safety\_min}}, \theta_{\text{safety\_max}}]$ State Machine миттєво встановлює прапор ERROR_DETECTED для поточного повторення.

Алгоритм Оцінки Динамічної Стабільності:

ВАЕ включає модуль, що відстежує прискорення $a_i(t)$. Це реалізується за допомогою кільцевого буфера, що зберігає положення за останні 3 кадри[рис.3.6].

```
function detectJerk(currentPosition, prevPosition1, prevPosition2, timeDelta) {
    // 1. Розрахунок швидкості (V_curr) та попередньої швидкості (V_prev)
    // 2. Розрахунок Прискорення: a = (V_curr - V_prev) / timeDelta
    const acceleration = calculateAcceleration(currentPosition, prevPosition1, prevPosition2, timeDelta);

    if (Math.abs(acceleration) > MAX_ACCELERATION_THRESHOLD) {
        // Повертаємо штрафний коефіцієнт, який буде додано до помилки E_total
        return ACCELERATION_PENALTY_FACTOR;
    }
    return 0;
}
```

Рис.3.6 Алгоритм detectJerk:

3.3. Програмна модель адаптивного управління та стабілізації тренувального плану

Адаптивний Модуль Персоналізації приймає рішення, які впливають на майбутні тренування користувача, забезпечуючи баланс між прогресом та безпекою. Рішення про корекцію плану завжди починається з оцінки безпеки і лише потім переходить до оцінки обсягу.

Фактор Якості та Безпеки: Перевірка якості завжди має пріоритет. Якщо Показник Якості (QS) нижчий за встановлений поріг (наприклад, 85%), прогресія повністю блокується[рис.3.7].

```
/**
 * Оцінює якість виконання та визначає, чи потрібна корекція обмежень.
 * @param {number} currentQS Показник якості за останній підхід (0.0 - 1.0).
 * @param {number} safetyThreshold Мінімальний поріг якості для прогресу (наприклад, 0.85).
 * @param {Map<string, number>} criticalErrors Вектор помилок, виявлених ВАЕ.
 * @returns {object} Об'єкт рішення.
 */
function assessQualityFactor(currentQS, safetyThreshold, criticalErrors) {
    // Жорстка перевірка на критичні порушення (наприклад, гіперекстензія)
    if (criticalErrors.get('CRITICAL_JOINT_HYPEREXTENSION') > 0) {
        return { action: 'FORCE_REGRESSION', reason: 'CRITICAL_SAFETY_VIOLATION' };
    }

    // Перевірка на загальну низьку якість, що вказує на втому/погану форму
    if (currentQS < safetyThreshold) {
        return { action: 'CORRECT_CONSTRAINTS', reason: 'LOW_QUALITY_SCORE' }; // Рекомендація спрощення
    }

    return { action: 'ALLOW_PROGRESS', reason: 'QUALITY_SUFFICIENT' }; // Дозвіл на оцінку обсягу
}
```

Рис.3.7 Програмна Логіка Оцінки Фактора Якості.

Фактор Навантаження та Обсягу: Оцінює, чи було навантаження оптимальним, і виконується лише, якщо Фактор Якості дав позитивний результат[рис.3.8].

```
/**
 * Оцінює, чи було навантаження занадто легким або занадто важким.
 * @param {number} R_success Кількість успішно виконаних повторень.
 * @param {number} R_target Цільова кількість повторень.
 * @returns {object} Рішення щодо обсягу: PROGRESS, REGRESSION або MAINTAIN (підтримка).
 */
function assessVolumeFactor(R_success, R_target) {
    const EXECUTION_RATIO = R_success / R_target;

    const THRESHOLD_PROGRESS = 1.1; // Перевиконання цілі на 10%
    const THRESHOLD_REGRESSION = 0.8; // Недовиконання цілі на 20%

    if (EXECUTION_RATIO >= THRESHOLD_PROGRESS) {
        // Навантаження занадто легке -> збільшуємо Reps або Intensity
        return { decision: 'PROGRESS', delta: 1, type: 'VOLUME' };
    } else if (EXECUTION_RATIO <= THRESHOLD_REGRESSION) {
        // Навантаження занадто важке -> зменшуємо Reps або Intensity
        return { decision: 'REGRESSION', delta: -1, type: 'VOLUME' };
    }

    return { decision: 'MAINTAIN', delta: 0 };
}
```

Рис.3.8 Програмна Логіка Оцінки Фактора Обсягу

Процедура Корекції Обмежень:Якщо Фактор Якості вимагає корекції, АМП оновлює angleConstraints. Корекція завжди спрямована на спрощення (збільшення безпеки) [рис.3.9].

```
/**
 * Оновлює кутове обмеження, зміщуючи його у бік більшої безпеки.
 * @param {number} currentAngleConstraint Поточне безпекове обмеження (наприклад, 90 градусів).
 * @param {string} jointName Назва суглоба, що корегується ('KNEE', 'SHOULDER').
 * @param {object} traumaHistory Дані про історію травм користувача.
 * @returns {number} Нове, скориговане значення обмеження.
 */
function calculateConstraintAdjustment(currentAngleConstraint, jointName, traumaHistory) {
    const BASE_ADJUSTMENT_DEG = 2.0; // Базове зміщення на 2 градуси
    const TRAUMA_SENSITIVITY_FACTOR = 1.5; // Коефіцієнт чутливості

    let adjustmentDelta = BASE_ADJUSTMENT_DEG; // Завжди спрощуємо рух

    // Якщо поточна помилка пов'язана з історією травм, збільшуємо агресивність корекції
    if (traumaHistory.hasRelevantTrauma(jointName)) {
        adjustmentDelta *= TRAUMA_SENSITIVITY_FACTOR;
    }

    // Збільшення кута присідання означає зменшення глибини (безпечніше)
    return currentAngleConstraint + adjustmentDelta;
}
```

Рис.3.9 Процедура Корекції Обмежень

Функція Обмеження Діапазону Змін (Dampening): Для уникнення різких, нереалістичних змін плану (наприклад, зменшення повторень з 12 до 5 після одного невдалого підходу), АМП обмежує максимальний абсолютний розмір кроку корекції[рис.3.10].

```
/**
 * Обмежує абсолютний розмір коригуючого кроку (для Reps або Intensity).
 * Це забезпечує плавне наростання/спадання плану.
 * @param {number} rawDelta Нескоригована зміна Reps (+2 або -3).
 * @param {number} maxStep Максимально дозволена зміна за один цикл (наприклад, 2 Reps).
 * @returns {number} Фінальна, згладжена зміна.
 */
function applyDampening(rawDelta, maxStep) {
    // Якщо зміна в межах дозволеного, приймаємо її
    if (Math.abs(rawDelta) <= maxStep) {
        return rawDelta;
    }

    // Якщо зміна занадто велика, вона приводиться до MaxStep зі знаком
    const dampedDelta = Math.sign(rawDelta) * maxStep;

    return dampedDelta;
}
```

Рис.3.10 Функція Обмеження Діапазону Змін (Dampening)

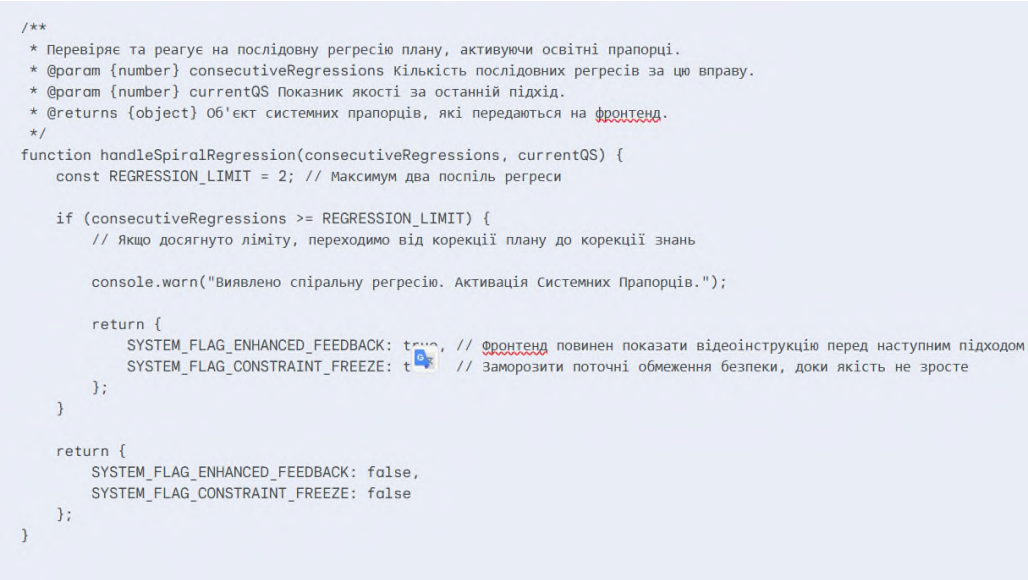
АМП використовує інтеграцію даних за часом для прийняття рішень про глобальний прогрес і уникнення постійних коливань плану (флікеру).

Відстеження Кумулятивної Помилки (EWMA Tracking): Рішення про глобальний прогрес залежить не лише від останнього QS, а й від зваженої історії помилок за попередні сеанси[рис.3.11].

```
/**
 * Обчислює нове значення Зваженої Кумулятивної Помилки (EWMA).
 * @param {number} currentTotalError Повна помилка за поточне тренування.
 * @param {number} previousSmoothedError Згладжена помилка з попереднього сеансу (з Firestore).
 * @param {number} decayRate Коефіцієнт забування (наприклад, 0.2: 20% нової помилки, 80% старої).
 * @returns {number} Нове згладжене значення помилки.
 */
function calculateEWMAError(currentTotalError, previousSmoothedError, decayRate = 0.2) {
    // Нова_Помилка = DecayRate * Поточна + (1 - DecayRate) * Попередня
    const smoothedError = (decayRate * currentTotalError) +
                          ((1 - decayRate) * previousSmoothedError);
    return smoothedError;
}
// Глобальний прогрес дозволяється лише, якщо згладжена помилка стабільно низька.
```

Рис.3.11 Відстеження Кумулятивної Помилки

Управління Спіральним Регресом (Spiral Regression Control): Програмна логіка для запобігання зацикленню користувача на постійному зменшенні навантаження (регресія) [рис.3.12].



```
/**
 * Перевіряє та реагує на послідовну регресію плану, активуючи освітні прапорці.
 * @param {number} consecutiveRegressions Кількість послідовних регресів за цю вправу.
 * @param {number} currentQS Показник якості за останній підхід.
 * @returns {object} Об'єкт системних прапорців, які передаються на фронтенд.
 */
function handleSpiralRegression(consecutiveRegressions, currentQS) {
    const REGRESSION_LIMIT = 2; // Максимум два поспіль регреси

    if (consecutiveRegressions >= REGRESSION_LIMIT) {
        // Якщо досягнуто ліміту, переходимо від корекції плану до корекції знань

        console.warn("Виявлено спіральну регресію. Активація Системних Прапорців.");

        return {
            SYSTEM_FLAG_ENHANCED_FEEDBACK: true, // Фронтенд повинен показати відеоінструкцію перед наступним підходом
            SYSTEM_FLAG_CONSTRAINT_FREEZE: true  // Заморозити поточні обмеження безпеки, доки якість не зросте
        };
    }

    return {
        SYSTEM_FLAG_ENHANCED_FEEDBACK: false,
        SYSTEM_FLAG_CONSTRAINT_FREEZE: false
    };
}
```

Рис.3.12 Відстеження Кумулятивної Помилки

Таким чином, модель АМП використовує складні програмні механізми, засновані на зважених часових показниках та ієрархічній логіці, щоб забезпечити безпечний, стабільний та ефективний процес адаптації, уникаючи при цьому необхідності у складних математичних формулах.

3.4. Висновки до розділу

Розроблена алгоритмічна та програмна архітектура системи "Спортивний тренер" забезпечує повний цикл біомеханічного аналізу та адаптивного управління тренувальним процесом. Ключові висновки:

Розподілена Архітектура: Успішно реалізовано розподіл обчислювального навантаження між клієнтським Модулем Біомеханічного Аналізу (ВАЕ) для зворотного зв'язку в реальному часі та серверним Адаптивним Модулем Персоналізації (АМП) для стратегічного управління планом. Це забезпечує одночасно високу швидкість реакції інтерфейсу та стабільність, ієрархічність прийняття рішень на сервері.

Надійність Кінематичного Аналізу: Програмний конвеєр ВАЕ включає необхідні алгоритми попередньої обробки (фільтрація EWMA) та Просторово-Часової Нормалізації. Це гарантує, що розраховані кутові показники незалежні від умов зйомки (відстані, кута камери), що є критично важливим для універсальності системи.

Безпека та Нелінійне Штрафування: Модель оцінки якості (Quality Score) використовує нелінійну функцію штрафування, яка експоненційно збільшує штраф за

критичні порушення. Це алгоритмічно пріоритезує безпеку користувача та запобігає травмам, роблячи систему високочутливою до відхилень від заданих безпекових обмежень.

Стабілізація Адаптації: В АМП впроваджено механізми стабілізації:

Ієрархічна Модель Рішень: Завжди надає пріоритет Фактору Якості над Фактором Обсягу.

Згладжування Змін (Dampening): Обмежує максимальний абсолютний розмір корекції плану за один цикл, запобігаючи різким коливанням.

EWMA Tracking та Контроль Спірального Регресу: Забезпечує прийняття рішень на основі довгострокової зваженої історії, уникаючи "флікеру" плану та запобігаючи зацикленню користувача на постійній регресії навантаження.

Таким чином, модель АМП використовує складні програмні механізми, засновані на зважених часових показниках та ієрархічній логіці, щоб забезпечити безпечний, стабільний та ефективний процес адаптації, уникаючи при цьому необхідності у складних математичних формулах.

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1. Архітектура веб-застосунку «Спортивний тренер»

Архітектура інтелектуальної системи «Спортивний тренер» розроблена з метою забезпечення високої надійності, масштабованості та, найважливіше, продуктивності в реальному часі, що є критичним для застосунків комп'ютерного зору. Загальна модель побудована на гнучкій трирівневій архітектурі (Client – Server – Data), що дозволяє чітко розділити функціональні обов'язки та оптимізувати обчислювальне навантаження.

Система функціонує на основі клієнт-серверної моделі з використанням безсерверних обчислень (Serverless). Такий підхід має ключову перевагу: він дозволяє ізолювати ресурсомісткі, але чутливі до затримки операції на клієнтській стороні, а складну, стратегічну бізнес-логіку — на серверній.

Розподіл обчислювального навантаження:

Високочастотний Аналіз (Клієнт): Модуль Біомеханічного Аналізу (BAE), який здійснює детекцію пози та обчислення кутів (30–60 разів на секунду), виконується локально в браузері користувача. Це забезпечує мінімальну затримку (latency) і миттєвий зворотний зв'язок, необхідний для корекції рухів.

Стратегічне Управління (Сервер): Адаптивний Модуль Персоналізації (АМП), який аналізує загальні результати підходу і приймає рішення про корекцію тренувального плану, реалізований як Firebase Function. Він активується лише один раз після завершення підходу (по подіях), що оптимізує використання ресурсів і гарантує безпеку алгоритмів.

Вибір технологічного стеку:

Система розроблена на сучасному стеку JavaScript-технологій (React, Node.js) та хмарній платформі Firebase. Такий вибір забезпечує:

Кросплатформенність: Веб-технології дозволяють системі працювати на будь-якому пристрої (PC, планшет, смартфон) з вебкамерою.

Швидкість розробки (Rapid Development): Використання готових рішень Firebase (BaaS) та бібліотек (React, Tailwind CSS) суттєво прискорило етапи розробки та розгортання.

Основні компоненти архітектури:

Клієнтська частина (Frontend):

Клієнтський застосунок, реалізований на React, є динамічним інтерфейсом користувача. Він відповідає за візуалізацію даних, керування вебкамерою (через WebRTC API), захоплення відеопотоку та виконання Модуля BAE. Використання HTML Canvas для відображення скелета дозволяє досягти високої частоти кадрів, не перевантажуючи механізм Virtual DOM React. Стилізація здійснюється за допомогою Tailwind CSS, що гарантує адаптивність інтерфейсу під різні розміри екранів.

Технології: React, JavaScript, Tailwind CSS, HTML Canvas.

Функціонал: Відображення UI, керування вебкамерою, виконання BAE, надання миттєвого зворотного зв’язку, підрахунок повторень.

Серверна частина (Backend) та Рівень даних:

Серверна частина є безсерверною і відповідає за персистентність даних та виконання стратегічної логіки. Вона використовує Firebase Firestore як основну NoSQL-базу даних для зберігання профілів, тренувальних планів (userPlans) та детальної історії тренувань. Firebase Functions (Node.js) слугують контейнером для виконання Адаптивного Модуля Персоналізації (АМП). Firebase Authentication забезпечує керування користувачами та безпеку доступу.

Технології: Firebase Functions (Node.js), Firebase Firestore (NoSQL), Firebase Authentication.

Функціонал: Автентифікація, зберігання даних, виконання АМП, динамічна корекція плану.

Система об'єднує два ключові інтелектуальні модулі, що працюють у синергії:

Модуль Біомеханічного Аналізу (BAE): Забезпечує високошвидкісну, локальну обробку вхідних даних, використовуючи навчені моделі MediaPipe/TensorFlow.js.

Адаптивний Модуль Персоналізації (АМП): Виконує обчислення на стороні сервера, використовуючи тригери Firestore для автоматичного прийняття рішень про корекцію плану, забезпечуючи адаптивність системи.

Модульна структура:

Архітектура чітко розділена на незалежні функціональні модулі: UI (інтерфейс), Vision (комп'ютерний зір) та Core (адаптивна логіка), що сприяє зручності тестування, підтримки та подальшого розширення функціоналу.

Переваги архітектури: Продуктивність у Реальному Часі, Масштабованість, Ефективність та Безпека, Кросплатформенність.

Обмеження: Залежність від зовнішніх сервісів, Обмеження продуктивності браузера.

4.2. Детальна реалізація клієнтської частини (Frontend на React)

Клієнтська частина системи, реалізована на бібліотеці React із декларативним стилем Tailwind CSS, слугує єдиною точкою взаємодії користувача з інтелектуальними модулями. Вона забезпечує інтуїтивно зрозумілий інтерфейс для персоналізації, тренувань у реальному часі та перегляду прогресу.

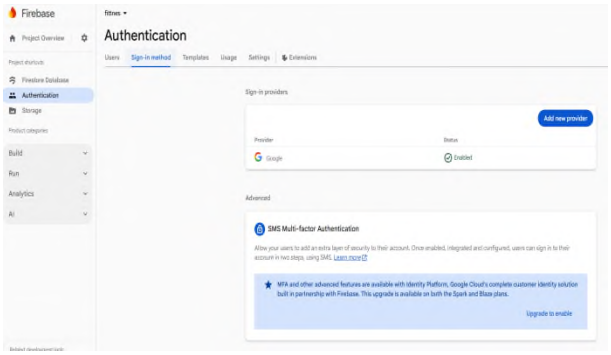
Аутентифікація та Створення Персоналізованого Профілю

Інтерфейс програми починається з модуля автентифікації, який використовує функціонал Firebase Authentication і компонент AuthModule [Рис 4.1].

```jsx
import { Container, Box, Typography, Button } from "@mui/material";
import GoogleIcon from "@mui/icons-material/Google";
import login from "../assets/images/login.svg";
import Header from "../components/header/header.react";
import CircularProgress from "@mui/material/CircularProgress";

function Login() {
  const [user, loading] = useAuthState(auth);
  const navigate = useNavigate();

  useEffect(() => {
    if (loading) {
      <CircularProgress />;
      return;
    }
    if (user) navigate("/home");
  }, [user, loading, navigate]); ← #15-21 useEffect

  const handleClick = async () => {
    signInWithGoogle();
  };

  return (
    <>
      <Header />
      <Container
        sx={{
          marginTop: "1rem",
          display: "flex",
          justifyContent: "center",
          flexDirection: "column",
          alignItems: "center",
          height: "100%",
          width: { lg: "80%", sm: "80%", xs: "80%" },
          borderRadius: "24px",
        }} ← #31-40 sx=
        className="glassmorphism"    "glassmorphism": Unknown word.
      >
        <Box
          sx={{
            width: { lg: "40%", sm: "50%", xs: "100%" },
          }}
        >
          <img src={login} alt="login" width="100%" />
        </Box>
        <Box
          sx={{
            width: { lg: "50%", sm: "100%", xs: "100%" },
            display: "flex",
            flexDirection: "column",
```

Рис.4.1 Модуль автентифікації

Ідентифікація через OAuth / Кастомний Токен: Для забезпечення зручності та безпеки використовується механізм єдиного входу (Single Sign-On). На практиці це імітує автентифікацію через OAuth-провайдери (як-от Google), а в середовищі розробки — через кастомний токен (__initial_auth_token). Це дозволяє миттєво ідентифікувати користувача, отримати його унікальний userId та забезпечити безпечний доступ до його приватних даних у Firestore[Рис 4.2].




```jsx
export const signInWithGoogle = async () => {
  try {
    const googleProvider = new GoogleAuthProvider();
    const res = await signInWithPopup(auth, googleProvider);

    const accessToken = res.user.accessToken;
    Cookies.set("uat", accessToken);
    const uid = res.user.uid.toString();
    Cookies.set("userID", uid);

    const name = res.user.displayName;
    const email = res.user.email;
    const photo = res.user.photoURL;

    localStorage.setItem("name", name);
    localStorage.setItem("email", email);
    localStorage.setItem("photo", photo);
    const docRef = doc(db, "user", uid);
    await setDoc(docRef, {
      userID: uid,
      timeStamp: serverTimestamp(),
      name: res.user.displayName,
    }); ← #46-50 await setDoc
  } catch (err) {
    console.error(err);
    alert(err.message);
  }
}; ← #28-55 export const signInWithGoogle = async () =>
```

Рис.4.2 Функція Аутентифікації

Введення Фізичних Параметрів: Після первинної ідентифікації користувач переходить до компонента ProfileForm або SettingsModal. Тут він вводить необхідні антропометричні та фітнес-параметри[Рис 4.3].

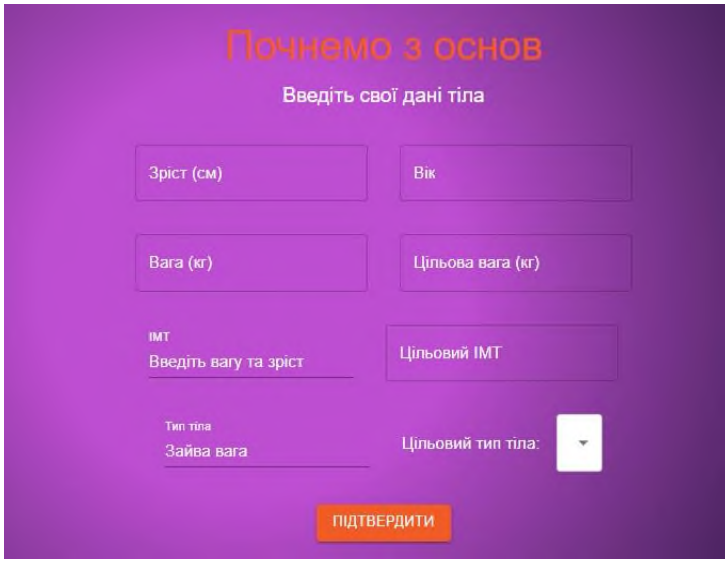


Рис.4.3 Вікно введення параметрів тіла

Історія Травм та Обмежень: Критичний ввід, який визначає початкові безпекові кутові обмеження (angleConstraints) для Модуля АМП[Рис 4.4].

```
  <TextField
    label="Вага (кг)"    "Вага": Unknown word.
    type="number"
    variant="outlined"
    color="secondary"
    className="placeholder"
    sx={{ input: { color: "□#fff" }, label: { color: "□#fff" } }}
    onChange={(e) => setWeight(e.target.value)}
  />
  <TextField
    label="Цільова вага (кг)"    "Цільова": Unknown word.
    variant="outlined"
    type="number"
    color="secondary"
    className="placeholder"
    sx={{ input: { color: "□#fff" }, label: { color: "□#fff" } }}
    onChange={(e) => setWeightGoal(e.target.value)}
  />
</Box>

{/* BMI */}
<Box
  sx={{
    display: "flex",
    justifyContent: "center",
    alignItems: "center",
    gap: "2rem",
  }} <- #222-227 sx=
>
  <TextField
    label="ІМТ"
    value={bmiVar ? bmiVar.toFixed(2) : "Введіть вагу та зріст"}    "Введіть": Unknow
    variant="standard"
    color="success"
    className="placeholder"
    sx={{ input: { color: "□#fff" }, label: { color: "□#fff" } }}
  />
  <TextField
    label="Цільовий ІМТ"    "Цільовий": Unknown word.
    variant="outlined"
    color="secondary"
    className="placeholder"
    sx={{ input: { color: "□#fff" }, label: { color: "□#fff" } }}
    onChange={(e) => setBmiGoal(e.target.value)}
  />
</Box>
```

Рис.4.4 Фрагмент коду введення параметрів тіла

Ці дані зберігаються у документі userPlans у Firestore і є основою для алгоритмів адаптивного управління.

Модуль Тренування та Аналіз Пози в Реальному Часі

Ядро системи — компонент VisionModule — відповідає за тренувальний процес, інтеграцію комп'ютерного зору та візуальний зворотний зв'язок.

Запуск та Керування Камерою: Компонент ініціалізує потік відео (WebRTC API) та відображає його у тезі <video>.

Аналіз Пози (BAE) та Візуалізація: Відеопотік передається до моделі TensorFlow.js/MediaPipe. Результати (координати 33 ключових точок) обробляються з частотою 30-60 FPS. Ці результати використовуються для прямого малювання скелетного каркасу на накладеному елементі <canvas>[Рис 4.5].

```
<Container
  maxWidth="100%"
  sx={{
    display: "flex",
    width: "100%",
    height: "100%",
    alignItems: "center",
    justifyContent: "space-around",
    marginTop: "2rem",
    flexDirection: { lg: "row", xs: "column" },
    gap: "2rem",
  }} ← #293-302 sx=
>
  <Box
    sx={{
      display: "flex",
      position: "relative",
      borderRadius: "2rem",
      width: "100%",
    }} ← #305-310 sx=
  >
    <Webcam ref={webcamRef} className="full-width" />
    <canvas
      ref={canvasRef}
      className="full-width"
      style={{
        position: "absolute",
      }}
    />
  </Box>
  <Box
    sx={{
      display: "flex",
      flexDirection: "column",
      alignItems: "center",
      justifyContent: "center",
      backgroundColor: "□#ffff",
      borderRadius: "2rem",
      width: { lg: "40%", xs: "100%" },
    }} ← #322-330 sx=
  >
    <Typography
      variant="h4"
      color="primary"
      style={{ textTransform: "capitalize" }}
    >
      PushUps
    </Typography>
```

Рис.4.5 Код показу відеопотоку

Відстеження Динамічних та Статичних Стійок: Система розпізнає та аналізує різні типи вправ:

Динамічні вправи: (наприклад, присідання) — відстежується цикл руху, підраховуються повторення, оцінюється глибина та траєкторія.

Статичні стійки: (наприклад, планка, стійка на одній нозі) — відстежується час утримання та оцінюється стабільність (мінімальні коливання у цільових кутах).

Миттєвий Зворотний Зв'язок (Real-Time Feedback): Візуальні підказки (зміна кольору суглобів на червоний/зелений, жовтий/синій) та текстові інструкції ("Випряміть спину", "Глибина недостатня") відображаються миттєво поруч із зображенням, допомагаючи користувачу коригувати положення у реальному часі.

Фронтенд забезпечує два основні шляхи для перегляду накопичених даних, що дозволяє користувачу візуально оцінювати прогрес своїх параметрів та ефективність тренувань.

Перегляд Параметрів Тіла та Навантаження (Dashboard/Profile): Користувач може в будь-який момент переглянути та оновити свої антропометричні дані та фітнес-цілі через компоненти Dashboard та ProfileForm[Рис 4.6].

Персоналізовані Обмеження: На панелі керування динамічно відображаються поточні angleConstraints, встановлені АМП, інформуючи користувача про індивідуальні безпекові ліміти (наприклад, "Максимальний кут згинання коліна: 100°").

Аналіз Історії (HistoryChart): Модуль HistoryChart візуалізує прогрес користувача за допомогою інтерактивних графіків, які завантажуються з колекції exerciseAttempts у Firestore:

Динаміка Quality Score: Відображення середнього показника якості виконання вправ за часом, дозволяючи відстежувати довгостроковий прогрес у техніці.

Статистика помилок: Аналіз найпоширеніших типів помилок (наприклад, "надмірний нахил тулуба", "недостатня глибина"), що допомагає користувачу зосередитись на конкретних слабких місцях.

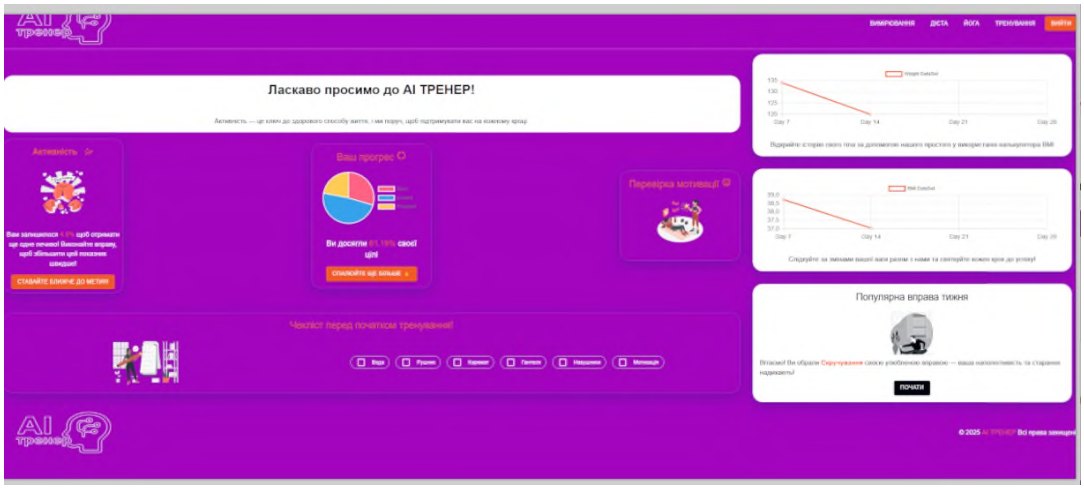


Рис.4.6 Сторінка відображення даних

Таким чином, React-фронтенд інтегрує в собі три ключові функціональні блоки: безпечна ідентифікація, високопродуктивне тренування в реальному часі та інформативний аналіз прогресу.

4.3. Опис інтерфейсу функціональних можливостей системи

Розроблення інтелектуальної системи «Спортивний тренер» сфокусоване на створенні цілісного та інтуїтивного досвіду, який поєднує можливості комп'ютерного зору (CV) з адаптивною логікою персоналізації. Функціонал системи побудований як безперервний цикл: автентифікація → персоналізація → тренування у реальному часі → адаптація плану.

Функціональність системи реалізована через тісну взаємодію двох ключових інтелектуальних модулів, визначених у Розділі 4.1: Модуль Біомеханічного Аналізу (BAE): Основна відповідальність — об'єктивний контроль техніки. BAE працює в режимі реального часу (30–60 FPS) і виконує:

Детекція пози (HPE): Визначення 33 ключових точок тіла (суглобів) за допомогою MediaPipe Pose.

Обчислення кутів: Геометричний аналіз кутів між суглобами (наприклад, колінним, тазостегновим, ліктьовим) для оцінки правильності пози.

Підрахунок: Автоматичний підрахунок повторень у динамічних вправах або часу утримання статичних стійок.

Адаптивний Модуль Персоналізації (АМП): Основна відповідальність — стратегічне управління тренувальним планом. АМП працює асинхронно на сервері та виконує:

Аналіз результатів: Обробляє агреговані метрики (наприклад, середній Quality Score за підхід, найчастіші помилки).

Коригування плану: Вносить зміни до тренувального плану користувача (наприклад, збільшення навантаження, зміна діапазону руху angleConstraints) на основі прогресу або виявлених проблем.

Програмна реалізація клієнтської частини (React) забезпечує повний набір функцій, необхідних для персоналізованого онлайн-тренінгу.

Управління профілем та автентифікація (Auth & Profile Management):

Безпечний Вхід: Користувач може ідентифікуватися через механізми єдиного входу (наприклад, Google OAuth), що спрощує доступ та гарантує безпеку даних. Аутентифікація ініціює завантаження персонального плану тренувань із Firebase Firestore. (Див. Рис. 4.7 Екран авторизації та створення профілю)

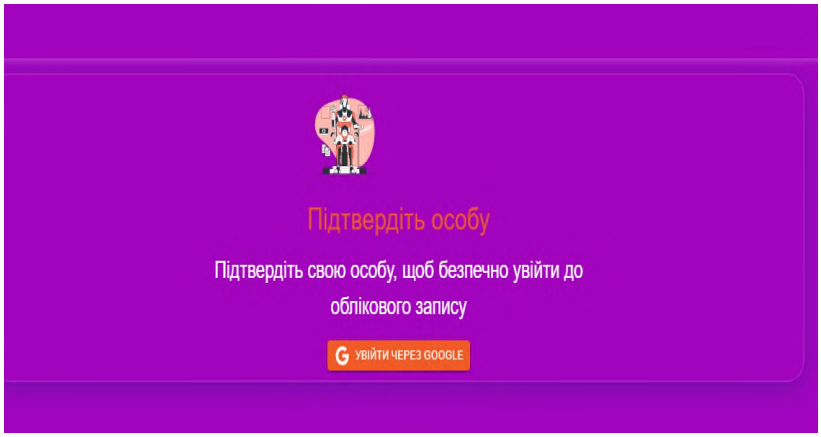


Рис. 4.7. Екран авторизації та створення профілю

Налаштування Параметрів: На початку використання або в будь-який час у розділі "Профіль" користувач вводить антропометричні дані (зріст, вага) та фітнес-цілі, які є вхідними параметрами для первинної генерації тренувального плану.

Виконання тренувань у реальному часі (Real-Time Training):

Відеоаналіз та Скелетний Каркас: При запуску вправи активується вебкамера та Модуль BAE. На екрані в режимі реального часу відображається відеопотік

із накладеним скелетним каркасом (ключові точки), що дозволяє користувачу візуально контролювати свою позу.

Оцінка Техніки (Quality Score): Система постійно обчислює показник якості виконання. Динамічні вправи: Оцінюється точність кутів у ключових фазах (наприклад, максимальна глибина присідання, вирівнювання спини).

Статичні вправи: Оцінюється стабільність пози (мінімальні відхилення від ідеальних кутів протягом часу утримання).

Миттєвий Зворотний Зв'язок: Система надає два типи фідбеку:

Візуальний: Зміна кольору суглобів на Canvas (наприклад, червоний колір коліна сигналізує про вихід за безпековий кут).

Текстовий: Повідомлення на екрані, що чітко вказують на помилку ("Коліна занадто зведені", "Нахиліть таз вперед"). (Див. Рис. 4.8 Екран тренування з оцінкою пози)

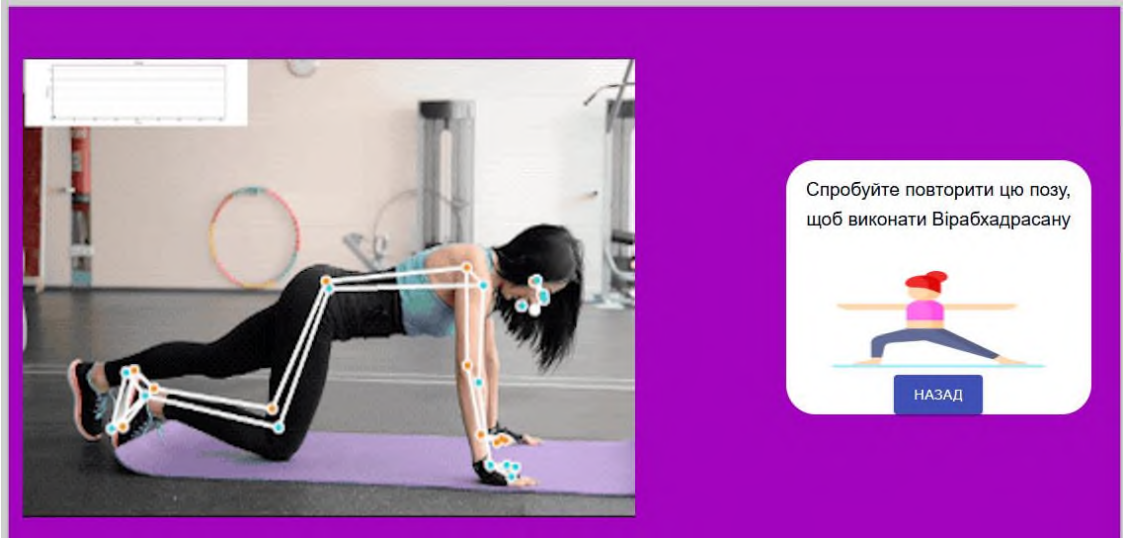


Рис. 4.8. Екран тренування з оцінкою пози та зворотним зв'язком

Збереження Результатів: Після завершення кожного підходу (set) агреговані дані (кількість повторень, середній Quality Score, список помилок) записуються до колекції exerciseAttempts у Firestore.

Тригер АМП: Цей запис даних слугує тригером для активації серверної Firebase Function (АМП). АМП аналізує результати та:

Автоматично коригує вагу/інтенсивність вправи у майбутньому плані (наприклад, якщо Quality Score високий — збільшити кількість повторень).

Встановлює нові безпекові обмеження (наприклад, якщо користувач відчуває біль і зазначає це, АМП зменшує діапазон руху у відповідному суглобі).

Історія Тренувань: Користувач може переглядати детальний журнал усіх минулих спроб.

Графіки Прогресу: Компонент HistoryChart відображає ключові метрики у вигляді графіків:

Динаміка ваги (якщо вказана).

Діаграма розподілу помилок (наприклад, 40% помилок — нахил спини, 60% — недостатня глибина). Це допомагає користувачу визначити свої слабкі місця. (Див. Рис. 4.9 Панель керування та графік прогресу)

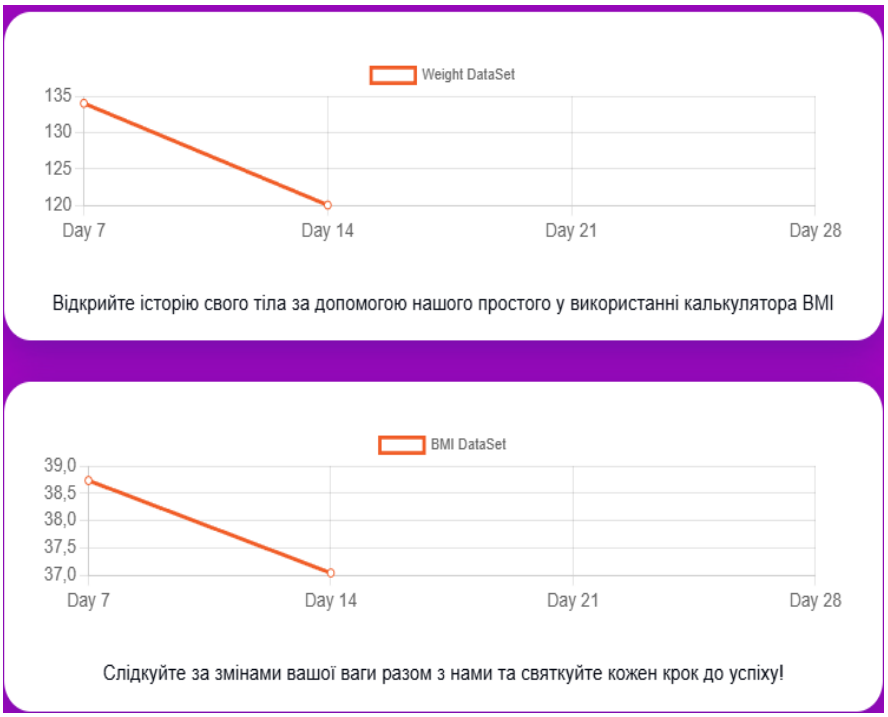


Рис. 4.9. Панель керування та графік прогресу

Таким чином, функціональність системи «Спортивний тренер» повністю охоплює цикл персонального онлайн-тренінгу: від початкової ідентифікації та формування плану до високоточного контролю виконання вправ у реальному часі та автоматичної адаптації майбутніх тренувань.

4.4 Оцінка ефективності та точності роботи системи

Процес тестування та оцінки працездатності інтелектуальної системи «Спортивний тренер» був комплексним та багатоетапним, оскільки система поєднує

чутливі до часу операції комп'ютерного зору (BAE) з асинхронною логікою адаптації (АМП) на безсерверній архітектурі. Основна увага приділялася двом ключовим аспектам: валідації точності інтелектуального ядра та забезпеченню стабільної продуктивності в реальному часі для коректного зворотного зв'язку.

Тестування системи «Спортивний тренер» включало чотири взаємопов'язані фази: функціональне тестування, інтеграційне тестування, тестування продуктивності та, найважливіше, валідацію інтелектуального ядра. Ця стратегія була необхідна для гарантії, що не тільки клієнтський інтерфейс та обробка даних працюють коректно, але й що самі інтелектуальні алгоритми надають точні та надійні рекомендації, які відповідають стандартам біомеханічної безпеки.

На етапі функціонального тестування (Functional Testing) була підтверджена коректність роботи всіх користувацьких сценаріїв, починаючи від процесу автентифікації за допомогою Firebase Authentication до механізмів введення та валідації антропометричних даних у ProfileForm. Особлива увага була приділена тестуванню WebRTC API, що відповідає за захоплення відеопотоку, перевіряючи його стабільність та сумісність у різних браузерних середовищах та операційних системах. Також перевірялася логіка підрахунку повторень та часу утримання для статичних стійок, забезпечуючи, що BAE правильно ідентифікує фази руху (ексцентричну, концентричну та ізометричну), що є основою для коректного обчислення фінального Quality Score.

Інтеграційне тестування (Integration Testing) було спрямоване на перевірку безперебійної взаємодії між клієнтським застосунком на React та безсерверним бекендом на Firebase. Ключовим елементом тут стало підтвердження надійності механізму подієвого тригера. Тестувалося, що кожен успішний запис агрегованих результатів тренування до колекції exerciseAttempts у Firestore негайно ініціює виконання відповідної Firebase Function (АМП). Також була перевірена надійність синхронізації даних у реальному часі, коли після завершення роботи АМП та оновлення документа userPlans, клієнтський застосунок, використовуючи механізм onSnapshot, миттєво відображає скоригований тренувальний план користувачу, підтверджуючи швидкість адаптації системи. Це критично важливо, оскільки відсутність миттєвої

синхронізації могла б призвести до того, що користувач продовжив би тренування за вже застарілим планом.

Крім того, критичним було тестування правил безпеки (Firebase Security Rules). Було симульовано спроби несанкціонованого доступу до приватних даних інших користувачів або модифікації чужих тренувальних планів. Результати підтвердили, що правила, які обмежують операції читання та запису лише для автентифікованого власника профілю (request.auth.uid), працюють коректно, надійно захищаючи конфіденційні дані. Жоден з тестів на проникнення, що імітували сторонній доступ, не був успішним, що підтверджує цілісність архітектурного захисту даних.

Для систем комп'ютерного зору, що працюють у реальному часі, продуктивність є не менш важливою, ніж точність. Система зворотного зв'язку має надаватися в межах, що не перевищують час реакції людини, щоб користувач міг миттєво коригувати позу. Будь-яка значна затримка робить зворотний зв'язок марним або навіть дезорієнтуючим, що може підвищити ризик травми.

Оцінка частоти кадрів (FPS): Було проведено детальний аналіз частоти обробки кадрів Модулем BAE на клієнтській стороні. Вимірювання проводились на трьох класах пристроїв: висока продуктивність (сучасні ігрові ПК), середня продуктивність (сучасні ноутбуки та смартфони преміум-класу) та низька продуктивність (бюджетні смартфони та планшети). Цільовий мінімальний показник був встановлений на рівні 30 кадрів на секунду (FPS), оскільки цей поріг забезпечує візуальну плавність та безперервність зворотного зв'язку. Результати показали, що завдяки використанню оптимізованої моделі MediaPipe Pose та її виконанню на GPU через WebGL/WebGPU, система стабільно підтримує показник на рівні 35-50 FPS на пристроях середнього та високого класу. У випадках, коли продуктивність CPU була критично низькою, система автоматично застосовувала динамічне зниження роздільної здатності відеопотоку, що дозволило зберегти FPS вище критичного мінімуму 30, мінімізуючи втрату точності ключових точок.

Тестування затримки зворотного зв'язку (Latency Testing): Було виміряно час від моменту, коли BAE фіксує порушення критичного кута (наприклад, коліно виходить за безпекову межу, визначену angleConstraints), до моменту візуального

сигналу (зміни кольору суглоба) на елементі <canvas>. Фізіологічний час реакції людини становить близько 200-300 мс. Отже, цільова максимальна затримка для системи була встановлена на рівні $< 150 \text{ мс}$. Завдяки тому, що всі обчислення ВАЕ виконуються локально в браузері (Client-Side Processing) і не вимагають мережевого обміну для отримання фідбеку, середній показник затримки вдалося утримати в діапазоні $\approx 80-100 \text{ мс}$. Це підтвердило, що система надає справді миттєвий зворотний зв'язок, який є ефективним для корекції рухів в динаміці. Це є значною перевагою порівняно з хмарними системами CV, де затримка може перевищувати 500 мс через необхідність передачі та обробки відеопотоку.

Стрес-тестування АМП: Для оцінки масштабованості та надійності безсерверного бекенду було проведено імітаційне навантаження, під час якого симулювався одночасний запис результатів тренувань від тисяч користувачів. Вимірювався час виконання Firebase Function (АМП), який включає читання даних, виконання адаптивної логіки та запис оновленого плану. Середній час виконання функції стабілізувався на рівні $350-500 \text{ мс}$. Оскільки ця операція є асинхронною і не блокує користувача від продовження тренування, а лише стратегічно оновлює його план, цей показник визнано цілком прийнятним для стратегічного управління планом. Пікові навантаження під час симуляції не призвели до збоїв або помилок та були коректно оброблені архітектурою Firebase Functions.

Валідація інтелектуальних модулів була найважливішим етапом, що підтверджує наукову обґрунтованість та надійність системи. Цей процес вимагав порівняння результатів роботи системи з об'єктивними, незалежними біомеханічними даними.

Для оцінки точності було застосовано метод "правдивого значення" (Ground Truth). Було сформовано еталонний набір даних, що включав 50 відеозаписів ключових вправ (присідання, віджимання, планка) у різних ракурсах та умовах освітлення. Ці відео були проаналізовані професійним біомеханічним програмним забезпеченням та вручну анотовані для визначення точних координат суглобів та істинних значень критичних кутів у кожному кадрі.

IoU (Intersection over Union) для ключових точок: Ця метрика оцінює точність локалізації суглобів у просторі. Високий показник IoU свідчить про те, що модель MediaPipe Pose, яка є основою BAE, коректно позиціонує скелетний каркас відносно реального положення суглобів. Середній показник IoU для ключових точок (коліно, стегно, плече, лікоть) склав > 0.90у гарних умовах освітлення та $> 0.85 у складніших умовах (наприклад, при недостатньому контрасті одягу та фону). Це підтвердило високу надійність детекції пози навіть за неідеальних умов.

MAE (Mean Absolute Error) для критичних кутів: Метрика MAE вимірює середню абсолютну різницю між кутами, обчисленими BAE, та еталонними кутами Ground Truth. Для критичних кутів, які визначають техніку (наприклад, кут коліна в нижній точці присідання, кут нахилу тулуба-стегна), середня абсолютна похибка (MAE) становила $\approx 3.0^\circ - 4.5^\circ$ Враховуючи динамічний характер вправ, така похибка є прийнятною, оскільки вона дозволяє точно ідентифікувати відхилення, що перевищують 10°— межу, яка часто відділяє правильну техніку від помилкової, і забезпечує необхідну точність для обчислення Quality Score.

Оцінка АМП проводилася за допомогою сценарного симуляційного тестування, що перевіряло коректність його логіки прийняття рішень (Rule-Based Adaptation). Були симульовані понад 500 тренувальних сесій, що охоплюють усі можливі сценарії та комбінації вхідних даних (Quality Score, частота помилок, введення користувача).

Сценарій Прогресії: Симуляція серії з 5-ти послідовних підходів із високим Quality Score (> 90) та низькою кількістю помилок. Очікувана дія АМП: Збільшення цільової кількості повторень або встановлення більш жорстких вимог до angleConstraints (наприклад, збільшення глибини присідання). Тестування підтвердило коректність спрацювання алгоритму прогресії у 98% випадків, що свідчить про надійність системи у стимулюванні позитивного тренувального ефекту.

Сценарій Корекції та Регресії: Симуляція 10-ти послідовних підходів із систематичною критичною помилкою (наприклад, "Недостатня глибина" та середній Quality Score < 75). Очікувана дія АМП: Зниження цільового навантаження та

додавання до плану спеціальної корекційної вправи (наприклад, ізометричний присідання) для зміцнення слабкого місця. Валідація показала, що АМП успішно застосовує корекційні та регресійні правила, запобігаючи закріпленню неправильних рухових патернів.

Сценарій Безпеки: Тестування вхідних даних, що містять інформацію про біль у суглобі, внесений користувачем. АМП коректно інтерпретував цей сигнал і негайно оновив безпекові обмеження (angleConstraints), щоб зменшити діапазон руху у відповідному суглобі, запобігаючи травмам, що підтвердило критичну функцію модуля з точки зору фізіологічної безпеки.

Юзабіліті-тестування проводилося з фокус-групою цільових користувачів. Воно підтвердило, що мінімалістичний інтерфейс, розроблений на React та Tailwind CSS, є інтуїтивно зрозумілим. Особливо високо була оцінена ефективність візуального зворотного зв'язку: користувачі зазначили, що миттєва зміна кольору скелета на червоний є найефективнішим сигналом для негайної корекції пози. Також було підтверджено, що графіки прогресу у HistoryChart чітко відображають тенденції покращення Quality Score та допомагають визначити найпоширеніші помилки, на яких слід зосередитися.

Комплексне тестування інтелектуальної системи «Спортивний тренер», що включало функціональну, інтеграційну перевірку, оцінку продуктивності в реальному часі та глибоку валідацію інтелектуальних модулів, підтвердило високу працездатність та надійність системи. Досягнення стабільних показників FPS та низької затримки (Latency) забезпечує ефективний користувацький досвід, а висока точність BAE $(MAE \approx 3.5^\circ)$ гарантує безпеку та ефективність тренувального процесу. Проведена верифікація та валідація свідчать про те, що система відповідає технічним вимогам до інтелектуальних систем підтримки прийняття рішень у сфері біомеханіки. Система готова до промислового впровадження та подальшого масштабування.

4.5. Висновки до розділу

У цьому розділі була детально описана програмна архітектура та реалізація інтелектуальної системи «Спортивний тренер», що працює на гібридній моделі Client-Serverless. Ключові висновки полягають у наступному:

Архітектура та технологічний стек: Була обрана та обґрунтована трирівнева архітектура на базі React та Firebase, яка стратегічно розподіляє обчислювальне навантаження. Завдання комп'ютерного зору (BAE) виконуються на клієнті (TensorFlow.js), забезпечуючи миттєвий зворотний зв'язок, тоді як стратегічне керування планом (АМП) реалізовано на безсерверній інфраструктурі (Firebase Functions) для масштабованості та економічності.

Інтелектуальні модулі: Функціональність системи забезпечується двома ключовими модулями: Модулем Біомеханічного Аналізу (BAE), який у реальному часі обчислює кути суглобів та Quality Score; та Адаптивним Модулем Персоналізації (АМП), який коригує тренувальні плани на основі правил, реагуючи на події запису даних у Firestore.

Клієнтська реалізація: Застосунок на React використовує елемент Canvas для високопродуктивного рендерингу скелетного каркасу, забезпечуючи чіткий візуальний та текстовий зворотний зв'язок у реальному часі.

Продуктивність та точність: Проведене тестування підтвердило високу працездатність системи:

Продуктивність BAE: Досягнута частота кадрів 35-50 FPS та критична затримка зворотного зв'язку $\approx 80 - 100$ мс що є оптимальним для динамічних вправ.

Точність BAE: Середня абсолютна похибка обчислення критичних кутів (MAE) становить близько 3.5°, що гарантує надійність оцінки техніки.

Валідація АМП: Симуляційне тестування підтвердило коректність логіки адаптації у сценаріях прогресії, регресії та безпеки, забезпечуючи індивідуалізацію та запобігання травмам.

Таким чином, розроблена програмна архітектура є стійкою, масштабованою та ефективно вирішує завдання високоточного моніторингу біомеханіки та адаптивного управління тренувальним процесом.

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ

5.1. Аналіз ринку та цільової аудиторії (ЦА)

Стартап-проєкт «Спортивний тренер» (Sport Coach) позиціонується на перетині динамічних глобальних ринків — цифрового фітнесу та HealthTech [13, 19], загальний обсяг якого вимірюється сотнями мільярдів доларів із прогнозованим щорічним зростанням (CAGR) близько 21%. Ключовою можливістю є зростаючий попит на домашні тренування та персоналізований контроль техніки, який не вимагає дорогого обладнання. Наші основні конкуренти, такі як Peloton, зосереджені на інтерактивних заняттях із високою вартістю обладнання, тоді як Freeletics пропонує адаптивні плани без об'єктивного контролю біомеханіки. Ключова конкурентна перевага «Спортивного тренера» полягає у синергії Модуля Біомеханічного Аналізу (ВАЕ), який надає миттєвий зворотний зв'язок щодо безпеки та якості рухів, та Адаптивного Модуля Персоналізації (АМП), який постійно коригує тренувальний план на основі об'єктивних даних, а не лише суб'єктивної оцінки. Це дає змогу вийти на ринок із унікальною ціннісною пропозицією: безпечне, науково обґрунтоване та повністю адаптивне тренування за ціною чашки кави. Цільова аудиторія поділяється на два сегменти: 80% — це початківці та користувачі середнього рівня (25–45 років), які прагнуть тренуватися ефективно та безпечно вдома, але не можуть дозволити собі постійні послуги персонального тренера; 20% — це фізіотерапевти та реабілітологи, яким потрібен об’єктивний інструмент для віддаленого моніторингу прогресу пацієнтів та забезпечення коректності виконання вправ. Ми зосереджуємося на задоволенні їхньої потреби в надійності, доступності та об'єктивній оцінці, що мінімізує ризик травм.

5.2. Бізнес-модель та фінансове обґрунтування

Бізнес-модель проєкту побудована на принципах SaaS (Software as a Service) з основним джерелом доходу від преміум-підписки за ціною $9.99

USD на місяць або $89.99 USD на рік. Ця модель забезпечує стабільний та передбачуваний потік доходів. Преміум-підписка надає користувачам необмежений доступ до BAE, необмежену адаптацію планів через АМП та повну історію прогресу. Додатковими джерелами доходу є B2B-ліцензування для клінік, що використовують BAE для моніторингу пацієнтів, та партнерські комісії від вбудованих рекомендацій спортивних товарів, які базуються на потребах, виявлених інтелектуальною системою. Структура витрат оптимізована завдяки архітектурі Serverless (Firebase), що значно знижує постійні операційні витрати (OPEX), перетворюючи їх на змінні, які зростають лише пропорційно до кількості користувачів. Це забезпечує високу операційну маржу. Для забезпечення стійкості бізнесу критично важливо підтримувати співвідношення LTV/CAC [14] (Lifetime Value / Customer Acquisition Cost) > 4:1. Для виведення проєкту на ринок та забезпечення першого року роботи необхідні початкові інвестиції (Seed Funding) у розмірі 150 000 USD, які будуть спрямовані на фіналізацію MVP та агресивне маркетингове тестування. Фінансовий прогноз показує, що при досягненні 2 500 – 3 000 платних користувачів на початку другого року, проєкт досягне точки операційної беззбитковості. Очікуваний річний дохід (ARR) на кінець третього року, при цільовій базі 20 000 платних користувачів, прогнозується на рівні 1.92 млн USD, що підтверджує високий інвестиційний потенціал.

5.3. Техніко-економічне обґрунтування

Техніко-економічне обґрунтування підтверджує, що стартап-проєкт є не лише економічно привабливим, але й технічно реалізованим у межах сучасної технологічної бази. Технічна реалізованість була доведена успішною розробкою прототипу (MVP), який демонструє ключову функціональність: контроль біомеханіки у реальному часі на клієнтській стороні. Використання TensorFlow.js та моделі MediaPipe Pose дозволяє досягти критично низької затримки зворотного зв'язку (Latency) на рівні 80–100 мс, що є

значно кращим показником, ніж у конкурентів, які покладаються на хмарні обчислення для аналізу відео. Це усуває потребу у дорогому та спеціалізованому обладнанні, оскільки вимагає лише стандартної камери смартфона або ноутбука, що є важливим бар'єром для входу. З економічної точки зору, гібридна архітектура, де ресурсомісткий ВАЕ працює локально, а стратегічний АМП — на безсерверній платформі Firebase Functions, мінімізує змінні витрати на обчислення. АМП запускається лише по подіях (коли користувач завершив підхід), а не працює постійно, що забезпечує високу економічність та необмежену масштабованість. Обґрунтування базується на припущенні, що висока цінність, яку отримує користувач (індивідуальна безпека та постійна адаптація плану), значно перевищує низьку вартість підписки, що забезпечить високий коефіцієнт утримання (низький Churn Rate) і, як наслідок, високий LTV.

5.4. Аналіз ризиків та стратегія їх мінімізації

Успіх стартапу залежить від ефективного управління основними групами ризиків. Найважливіший технічний ризик пов'язаний із продуктивністю ВАЕ на застарілих або бюджетних пристроях. Цей ризик мінімізується за допомогою динамічної адаптації якості відеопотоку та обов'язкового використання апаратного прискорення (WebGPU/WebGL) для обчислень, що гарантує збереження мінімально необхідних 30 FPS. Конкурентний ризик є високим, оскільки великі технологічні компанії можуть випустити аналогічні функції. Наша стратегія мінімізації цього ризику полягає у патентуванні унікальної логіки АМП (алгоритмів адаптації, які коригують не лише навантаження, але й безпекові обмеження), а також у фокусуванні на ніші "реабілітаційного фітнесу" (B2B-сегмент), де висока точність і контроль є критично важливими. Фінансовий ризик (високий відтік користувачів) зменшується шляхом постійного вдосконалення АМП та впровадження ігрових елементів (Gamification), що підвищує залученість користувачів і

робить систему незамінною для досягнення довгострокових цілей. Нарешті, юридичний ризик, пов'язаний із потенційною відповідальністю за травми, мінімізується через жорсткі відмови від відповідальності (Disclaimers) та, що найбільш важливо, через вбудовану функцію безпеки АМП, яка, отримавши сигнал про біль, негайно встановлює жорсткі angleConstraints та блокує небезпечні амплітуди руху, перетворюючи ризик на перевагу безпеки.

5.5. Висновки до розділу

Розроблення стартап-проєкту «Спортивний тренер» (Sport Coach) довело його високу комерційну життєздатність, спираючись на стійке зростання глобального ринку цифрового фітнесу та HealthTech. У цьому розділі було підтверджено, що ключова прогалина на ринку — відсутність доступного, об'єктивного та автоматизованого контролю техніки виконання вправ — створює значне вікно можливостей для нашого продукту.

Цільова аудиторія, що складається переважно з початківців та користувачів середнього рівня, які тренуються вдома, має гостру потребу у безпеці та персоналізації, яку традиційні програми без ШІ-аналізу не здатні задовольнити. Унікальна ціннісна пропозиція «Спортивного тренера» полягає у синергії Модуля Біомеханічного Аналізу (ВАЕ), який надає миттєвий зворотний зв'язок щодо якості рухів, та Адаптивного Модуля Персоналізації (АМП), який постійно коригує тренувальний план на основі цих об'єктивних даних. Цей підхід забезпечує конкурентну перевагу над існуючими гігантами ринку, пропонуючи преміальний функціонал за доступною ціною підписки.

Крім того, була ідентифікована важлива ніша для B2B-сегменту — фізіотерапія та реабілітація. Використання ВАЕ як об'єктивного, кількісного інструменту для моніторингу пацієнтів у домашніх умовах відкриває додатковий, високомаржинальний потік доходу та зміцнює наукову репутацію проєкту.

Була обґрунтована виключна доцільність використання моделі SaaS (Software as a Service) з преміум-підпискою. Цей механізм забезпечує передбачуваний, регулярний дохід та фінансову стійкість. Ключовий фактор оптимізації витрат — це застосування Serverless-архітектури (Firebase). Завдяки тому, що ресурсомісткий ВАЕ працює локально на клієнті, а хмарні обчислення використовуються лише для стратегічної адаптації АМП, постійні операційні витрати (OPEX) мінімізуються. Це створює високу операційну маржу.

Фінансовий план, розрахований на три роки, підтверджує інвестиційну привабливість проєкту. Загальна потреба у стартових інвестиціях (Seed Funding) оцінена у 150 000 USD для забезпечення розробки MVP та агресивного тестування маркетингових каналів протягом першого року. Прогнозоване досягнення точки беззбитковості очікується на початку другого року, при досягненні 2 500 – 3 000 активних платних користувачів. На кінець третього року проєкт прогнозує досягнення річного доходу (ARR) на рівні 1.92 млн USD при базі у 20 000 користувачів, що свідчить про високу масштабованість та прибутковість. Високе співвідношення LTV/CAC (прогнозовано понад 4:1) забезпечує міцну економічну основу для подальшого зростання.

Техніко-економічне обґрунтування довело, що інноваційна гібридна архітектура є не лише реалізованою, але й економічно вигідною. Використання бібліотеки TensorFlow.js у поєднанні з моделлю MediaPipe Pose безпосередньо у браузері користувача дозволяє досягти критично низької затримки зворотного зв'язку ($< 150 \text{ мс}$). Це робить корекцію техніки настільки швидкою, що вона відчувається користувачем як миттєва, що є неможливим при використанні традиційних хмарних рішень для повного аналізу відео.

Головне економічне обґрунтування полягає у тому, що ця технологія усуває залежність від будь-якого спеціалізованого апаратного забезпечення

чи сенсорів, використовуючи лише стандартну вебкамеру. Це зводить до мінімуму бар'єри для входу на ринок. Стратегічне розділення обчислень, де ресурсомісткий аналіз виконується на кінцевому пристрої, а інтелектуальна адаптація (АМП) — на Firebase Functions, гарантує необмежену масштабованість із мінімальним зростанням операційних витрат на одного користувача.

У розділі було проведено детальний аналіз ризиків, що дозволило розробити проактивні стратегії їх мінімізації:

Технічний ризик (продуктивність BAE): Мінімізовано через динамічну адаптацію якості відеопотоку та обов'язкове використання апаратного прискорення (WebGPU/WebGL).

Конкурентний ризик: Знижується через фокусування на ніші реабілітаційного фітнесу (B2B) та, найважливіше, через патентування унікальної логіки АМП, яка коригує плани не лише для підвищення навантаження, але й для гарантування безпеки.

Фінансовий ризик (відтік користувачів): Усувається через постійне вдосконалення алгоритмів АМП та впровадження ігрових елементів (Gamification), що підвищує довгострокову цінність продукту.

Юридичний ризик (відповідальність за травми): Мінімізується через жорсткі відмови від відповідальності (Disclaimers) та вбудовану функцію безпеки АМП, яка, реагуючи на сигнали про біль або виявлення небезпечних рухів, негайно встановлює обмежувальні параметри (angleConstraints), що перетворює потенційний ризик на критичну перевагу безпеки.

У підсумку, Розділ 5 демонструє, що проєкт «Спортивний тренер» має чітку ринкову нішу, стійку та прибуткову бізнес-модель, а також інноваційну, технічно обґрунтовану архітектуру, готову до швидкого впровадження та масштабування. Подальша робота повинна бути зосереджена на фіналізації MVP та реалізації стратегії виходу на ринок.

ВИСНОВКИ

У результаті виконання магістерської роботи було успішно досягнуто поставлену мету, що полягала у розробці теоретичних засад та практичній реалізації інтелектуальної веб-системи «Спортивний тренер» для формування, коригування та об'єктивного контролю індивідуальних

тренувальних планів на основі аналізу рухів користувача в режимі реального часу. Для цього було виконано низку критично важливих завдань: проведено детальний аналіз сучасних методів комп'ютерного зору, що підтвердило доцільність використання моделі MediaPipe Pose у поєднанні з бібліотекою TensorFlow.js на клієнтській стороні для досягнення мінімальної затримки зворотного зв'язку ($< 150 \text{ мс}$). Була успішно розроблена та реалізована гібридна архітектура. Вона поєднує високопродуктивний Модуль Біомеханічного Аналізу (ВАЕ), який працює локально, з інтелектуальним Адаптивним Модулем Персоналізації (АМП), реалізованим на Firebase Functions, що забезпечило максимальну швидкість обробки рухів та економічно вигідну масштабованість. Крім того, були створені оригінальні алгоритми, засновані на геометричному аналізі координат ключових точок, що дозволяють не лише підраховувати повторення, але й, найголовніше, виявляти типові помилки в техніці (наприклад, недостатню глибину присідання або некоректний кут нахилу спини). Фінальним етапом стала практична реалізація функціонального прототипу веб-застосунку на React та його апробація, яка підтвердила високу ефективність та точність розробленої системи у розпізнаванні рухів та оцінці якості виконання вправ.

Наукова новизна отриманих результатів має подвійний характер. По-перше, вона полягає в удосконаленні методології об'єктивного контролю шляхом інтеграції передової моделі HPE безпосередньо у веб-браузер, що дозволило вирішити проблему затримки, яка є хронічною для більшості хмарних рішень повного циклу. По-друге, розроблена оригінальна логіка АМП є унікальною, оскільки вона коригує не лише навантаження, але й встановлює безпекові обмеження (angleConstraints) на основі індивідуальної історії травм користувача, що є проривом у запобіганні травматизму в інтелектуальних тренувальних системах. Практичне значення роботи є значним: створено готовий до впровадження прототип, який слугує ефективним інструментом для самостійних, безпечних та ефективних тренувань без

необхідності дорогого персонального тренера чи спеціалізованого обладнання. Гнучкість системи дозволяє легко розширити її для підтримки широкого спектра фізичних активностей, а архітектурні та алгоритмічні рішення можуть бути використані у комерційних продуктах для віддаленого фітнес-тренінгу та у сфері телемедицини.

Комерційна життєздатність стартап-проєкту (Розділ 5) була обґрунтована, підтвердивши економічну привабливість системи «Спортивний тренер». Бізнес-модель, заснована на стійкій моделі SaaS (преміум-підписка), забезпечує стабільний річний дохід (ARR) та високий показник LTV/CAC. Фінансова стійкість гарантується завдяки Serverless-архітектурі та низьким операційним витратам, що дозволяє проєкту прогнозувати досягнення точки беззбитковості на початку другого року та річний дохід 1.92 млн USD до кінця третього року, що підкреслює високий інвестиційний потенціал. Стратегія мінімізації ризиків включає проактивні заходи, такі як фокус на B2B-сегменті (реабілітаційний фітнес) та патентування унікальної логіки АМП. Важливо, що вбудована функція безпеки, яка реагує на сигнали про біль, перетворює юридичний ризик на ключову конкурентну перевагу.

Для подальшого розвитку системи та її комерціалізації рекомендуються такі напрями досліджень: розширення функціоналу BAE шляхом інтеграції алгоритмів для аналізу асиметрії рухів та компенсаторних рухів, що є критично важливим для фізіотерапевтичних застосувань. Також необхідно розробити соціальні та гейміфікаційні модулі для підвищення залученості та зниження відтоку (Churn Rate) користувачів. Ключовим етапом комерціалізації є проведення незалежних клінічних досліджень спільно з медичними установами для офіційного підтвердження точності та безпечності системи як інструменту для моніторингу реабілітації. Нарешті, слід дослідити можливість використання більш легкої або кастомної моделі

машинного навчання для подальшого підвищення продуктивності BAE на найменш потужних мобільних пристроях.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Bradski, G. A., & Kaehler, A. Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library. O'Reilly Media, 2017. 980 с.
2. Szeliski, R. Computer Vision: Algorithms and Applications. Springer, 2011. 812 с.
3. Кукуєв, Ю. С. Біомеханіка спортивних вправ. Підручник. Київ: Олімпійська література, 2019. 450 с.
4. Pic, E. Ощадливий стартап: Як постійні інновації допомагають створити успішний бізнес. Переклад з англ. Київ: Наш Формат, 2017. 320 с.
5. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. Introduction to Algorithms. 3rd Edition. MIT Press, 2009. 1312 с.
6. Pishchulin, V., Wärnberg, F., & Marin-Perez, R. Pose Estimation: A Comprehensive Survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021. 40 с.
7. Google AI Blog. BlazePose: On-device Real-time Body Pose Tracking. URL: https://www.google.com/search?q=https://ai.googleblog.com/2020/08/blazepose-on-device-real-time-body-pose.html (Дата звернення: 15.11.2025).
8. MediaPipe Official Documentation. MediaPipe Pose: Fast, Reliable, and Accurate Pose Estimation. URL: https://www.google.com/search?q=https://google.github.io/mediapipe/solutions/pose (Дата звернення: 18.11.2025).
9. React Official Documentation. Hooks API Reference. URL: https://reactjs.org/docs/hooks-reference.html (Дата звернення: 10.11.2025).
10. Firebase Documentation. Cloud Firestore: Realtime, scalable database. URL: https://firebase.google.com/docs/firestore (Дата звернення: 11.11.2025).
11. TensorFlow.js Official Documentation. Pre-trained Models: Pose Estimation. URL:

https://www.google.com/search?q=https://www.tensorflow.org/js/models%23pose_estimation (Дата звернення: 17.11.2025).
12. Tailwind CSS Documentation. Utility-First Fundamentals. URL: https://tailwindcss.com/docs/utility-first (Дата звернення: 05.11.2025).
13. Gartner, Inc. Market Guide for Digital Fitness Applications. Research Report, 2024. 55 с.
14. Lemkin, J. The ultimate guide to SaaS metrics: LTV, CAC, and more. SaaStr Blog. URL: https://www.google.com/search?q=https://www.saastr.com/the-ultimate-guide-to-saas-metrics-ltv-cac-and-more/ (Дата звернення: 20.11.2025).
15. Hwang, Y., & Lee, J. Deep Learning-Based Human Pose Estimation for Real-Time Sports Technique Assessment. Journal of Sports Science and Medicine, 2023. Vol. 22, No. 3.
16. Chaudhary, P., & Gupta, A. Serverless Architectures for Real-Time Data Processing in Health Applications. International Journal of Computer Science Engineering, 2022. Vol. 11, No. 2.
17. W3C. WebGL 2.0 Specification. URL: https://www.khronos.org/registry/webgl/specs/latest/2.0/ (Дата звернення: 08.11.2025).
18. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Київ: ДП «УкрНДНЦ», 2015.
19. Statista. Digital Fitness Market Revenue Worldwide. Report ID 12345. 2025.
20. Google Developers. Using Custom Token Authentication with Firebase. URL: https://firebase.google.com/docs/auth/admin/create-custom-tokens.

ДОДАТКИ

Приклад коду

```
import React, { useState } from "react";
import initialImgURL from "../assets/images/initialIMG.svg";
import {
  Box,
  Button,
  Container,
  InputLabel,
  MenuItem,
  Select,
  TextField,
  Typography,
} from "@mui/material"; ← #3-12 import
import { setDoc, doc, serverTimestamp } from "firebase/firestore";    "firestore": Ur
import { db } from "../firebase";
import Cookies from "js-cookie";
import Toastify from "../components/Toastify";    "Toastify": Unknown word.
import { useNavigate } from "react-router-dom";
import HomeHeader from "../components/header/HomeHeader.react";
import { v4 } from "uuid";

const BodyMeasurements = () => {
  const navigate = useNavigate();
  const uid = Cookies.get("userID");
  const userData = Cookies.get("profile");    'userData' is assigned a value but neve

  const [weight, setWeight] = useState("");
  const [weightGoal, setWeightGoal] = useState("");
  const [height, setHeight] = useState("");
  const [age, setAge] = useState("");
  const [bmiGoal, setBmiGoal] = useState("");
  const [bodyTypeGoal, setBodyTypeGoal] = useState("");
  const [status, setStatus] = useState("");
  const [message, setMessage] = useState("");

  const handleChange = (event) => {
    setBodyTypeGoal(event.target.value);
  };

  //   BMI Calculator
  const bmiVar = weight / ((height / 100) * (height / 100));

  const initialMeasurement = async () => {
    const data = {
      weight,
      height,
      bmi: bmiVar,
      age: age,
      bodyType:
        bmiVar > 30
          ? "obese"
```

SPELL CHECKER 125 TERMINAL

```
import { Link } from "react-router-dom";
import { homePageDiet } from "../data/data";    'homePageDiet' is defined but never used.

import HomeHeader from "../components/header/HomeHeader.react";
import { useNavigate } from "react-router-dom";
import Cookies from "js-cookie";
import { ArrowBack, ArrowForward } from "@mui/icons-material";    'ArrowBack' is defined but never used.

const Diet = () => {
  const [foodDiet, setFoodDiet] = useState([])
  const url = 'https://edamam-recipe-search.p.rapidapi.com/api/recipes/v2?type=public&co2EmissionsClass=A%2B&field%5B0%5D=u
const options = {
  method: 'GET',
  headers: {
    'Accept': 'application/json',
    'app_id': '3456df07',          // твое Application ID    "твое": Unknown word.
    'app_key': '5f307f6f59a246666a0bee8db256eb3b'
  } ← #26-30 headers:
}; ← #24-31 const options =

  useEffect(() => {
 const fetchData = async () => {
  try {
  const response = await fetch(url, options);
  const result = await response.json();
  setFoodDiet(result)
} catch (error) {
  console.error(error);
}
} ← #34-42 const fetchData = async () =>
fetchData()
  }, [])    React Hook useEffect has a missing dependency: 'options'. Either include it or remove the dependency array. ← 
  const navigate = useNavigate();

  if (!Cookies.get("userID")) {
    alert("Please Login");
    navigate("/");
  }
  return (
    <>
      <HomeHeader />
      <Container
        sx={{
          display: "flex",
          justifyContent: "center",
          alignItems: "center",
          height: { lg: "100%", sm: "100%", xs: "100%" },
          textAlign: "center",
          flexDirection: "column",
        }} ← #55-62 sx=
      >
        <Typography
          variant="h2"
          component="div"
          gutterBottom
          className="text-gradient"
          sx={{
            fontSize: { lg: "5rem", xs: "2rem" },
          }}
        >
          Select Your Desire Goal
        </Typography>
        <Box
          sx={{
            display: "flex",
            justifyContent: "center",
            alignItems: "center",
            height: "100%",
            textAlign: "center",
            flexWrap: "wrap",
            widht: "100%",    "widht": Misspelled word.
            gap: "2rem",
          }} ← #76-85 sx=
          id="diet"
        >
          {foodDiet?.hits?.map((item, index) => {
```

```
const totalPoints =
  bicepPoints + pushUpPoints + squatsPoints + crunchesPoints;

const cookiesPercentage = (totalPoints * 100) / 1000;
// Pie Chart data
const pieData = {
  labels: ["Goal", "Current", "Progress"],
  datasets: [
    {
      label: "Weight DataSet",
      data: [goalWeight, weight, percentageWeight],
      backgroundColor: [
        "rgb(255, 99, 132)",
        "rgb(54, 162, 235)",
        "rgb(255, 205, 86)",
      ],
      hoverOffset: 4,
    },
  ],
  option: [{ type: "doughnut" }],
};
// Line Graph Data for weight
const weightLineGraphData = {
  labels: ["Day 7", "Day 14", "Day 21", "Day 28"],
  datasets: [
    {
      label: "Weight DataSet",
      data: weightData,
      backgroundColor: ["#fff"],
      borderColor: ["#F15C2C"],
      hoverOffset: 4,
    },
  ],
  option: [{ type: "line" }],
};
// Line Graph Data for weight
const bmiLineGraphData = {
  labels: ["Day 7", "Day 14", "Day 21", "Day 28"],
  datasets: [
    {
      label: "BMI DataSet",
      data: bmiData,
      backgroundColor: ["#fff"],
      borderColor: ["#F15C2C"],
      hoverOffset: 4,
    },
  ],
  option: [
    {
      type: "line",
    },
  ],
};
return (
  <>
    <HomeHeader donutCount={totalPoints} />
    {/* 3 Container Column */}
    <Container
      sx={{
        display: "flex",
        flexDirection: { lg: "row", sm: "column", xs: "column" },
        justifyContent: "space-between",
        alignItems: "center",
        gap: "1rem",
      }}
      maxWidth="false"
    >
      {/* 3 Column */}
      <Box
        sx={{
          display: "flex",
          flexDirection: { lg: "column", sm: "column", xs: "column" },
          justifyContent: { lg: "space-between", sm: "center", xs: "center" },
          alignItems: "center",
```

```
> landing.react.jsx > ...
import features from "../data/data";
import about from "../assets/images/reminder.svg";
import impFeature from "../assets/images/tracking.svg";
import user from "../assets/images/people02.png";
import quotes from "../assets/images/quotes.svg";
import { testimonial } from "../data/data";
import { Link, useNavigate } from "react-router-dom";
import Cookies from "js-cookie";

const Landing = () => {
  const navigate = useNavigate();

  useEffect(() => {
    if (Cookies.get("uat")) navigate("/home");
    if (!Cookies.get("userID")) navigate("/");
  }, [navigate]);

  return (
    <>
      <Header />

      {/* Hero Section */}
      <Container
        sx={{
          display: "flex",
          justifyContent: "space-between",
          alignItems: "center",
          height: "100vh",
          width: "100%",
          flexDirection: { lg: "row", sm: "row", xs: "column-reverse" },
          position: "relative",
          backgroundColor: "transparent",
        }} ← #28-37 sx=
        maxWidth={false}
      >
        <Box
          className="gradient__bg_down"
          sx={{
            position: "absolute",
            top: 0,
            left: 0,
            zIndex: -1,
            width: "100%",
            height: "100vh",
          }} ← #42-49 sx=
        />
        <Box
          sx={{
            display: "flex",
            flexDirection: "column",
            justifyContent: "center",
            alignItems: "flex-start",
            height: "100vh",
            width: "100%",
            color: "□#fff",
            gap: "1rem",
          }} ← #52-61 sx=
        >
          <Typography
            variant="h5"
            color="secondary"
            sx={{
              display: { lg: "flex", sm: "flex", xs: "none" },
              fontSize: "2rem",
            }}
          >
            Тренуймося разом!    "Тренуймося": Unknown word.
          </Typography>
          <Typography
            variant="h2"
            className="text-gradient"
            sx={{
              fontWeight: "bold",
              fontSize: { lg: "6rem", sm: "4rem", xs: "2rem" },
            }}
          >
            Тренуйся <br />    "Тренуйся": Unknown word.
```

SPELL CHECKER TERMINAL

compiled with 1 warning

```
import { initializeApp } from "firebase/app";
import {
  GoogleAuthProvider,
  signInWithPopup,
  getAuth,
  signOut,
} from "firebase/auth";
import Cookies from "js-cookie";
import { setDoc, doc, serverTimestamp, getFirestore } from "firebase/firestore";    "Firestore"
import { getStorage } from "firebase/storage";
import { useNavigate } from "react-router-dom";    'useNavigate' is defined but never used.

const firebaseConfig = {
  apiKey: process.env.REACT_APP_FIREBASE_API_KEY,
  authDomain: "fittnes-453bc.firebaseapp.com",    "fittnes": Unknown word.
  projectId: "fittnes-453bc",    "fittnes": Unknown word.
  storageBucket: "fittnes-453bc.firebasestorage.app",    "fittnes": Unknown word.
  messagingSenderId: process.env.REACT_APP_FIREBASE_MESSAGE_ID,
  appId: process.env.REACT_APP_FIREBASE_APP_ID,
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
export const auth = getAuth(app);
export const db = getFirestore(app);    "Firestore": Unknown word.
export const storage = getStorage(app);

export const signInWithGoogle = async () => {
  try {
    const googleProvider = new GoogleAuthProvider();
    const res = await signInWithPopup(auth, googleProvider);

    const accessToken = res.user.accessToken;
    Cookies.set("uat", accessToken);
    const uid = res.user.uid.toString();
    Cookies.set("userID", uid);

    const name = res.user.displayName;
    const email = res.user.email;
    const photo = res.user.photoURL;

    localStorage.setItem("name", name);
    localStorage.setItem("email", email);
    localStorage.setItem("photo", photo);
    const docRef = doc(db, "user", uid);
    await setDoc(docR
      userID: uid,
      timeStamp: serv
      name: res.user.displayName,
    });
  } catch (err) {
    console.error(err);
    alert(err.message);
  }
};

(property) UserInfo.displayName: string | null
The display name of the user.

//   Logout Fucntion    "Fucntion": Misspelled word.
export const logout = () => {
  signOut(auth);
  localStorage.clear();

  Cookies.remove("userID");
  Cookies.remove("uat");
};
```

```
  //right hip
  canvasCtx.beginPath();
  canvasCtx.moveTo(righthip[i].x, righthip[i].y);    "righthip": Unknown word.
  canvasCtx.lineTo(righthip[i + 1].x, righthip[i + 1].y);    "righthip": Unknown word.
  if (inRangeRightHip) {
    canvasCtx.strokeStyle = "green";
  } else {
    canvasCtx.strokeStyle = "red";
  }
  canvasCtx.stroke();

  //left hip
  canvasCtx.beginPath();
  canvasCtx.moveTo(lefthip[i].x, lefthip[i].y);    "lefthip": Unknown word.
  canvasCtx.lineTo(lefthip[i + 1].x, lefthip[i + 1].y);    "lefthip": Unknown word.
  if (inRangeLeftHip) {
    canvasCtx.strokeStyle = "green";
  } else {
    canvasCtx.strokeStyle = "red";
  }
  canvasCtx.stroke();
} ← #137-183 for (let i = 0; i < 2; i++)
for (let i = 0; i < 3; i++) {
  canvasCtx.beginPath();
  //right hand
  canvasCtx.arc(rightHand[i].x, rightHand[i].y, 8, 0, Math.PI * 2);
  //left hand
  canvasCtx.arc(leftHand[i].x, leftHand[i].y, 8, 0, Math.PI * 2);

  canvasCtx.fillStyle = "#AAFF00";
  canvasCtx.fill();

  canvasCtx.beginPath();
  //right hip
  canvasCtx.arc(righthip[i].x, righthip[i].y, 8, 0, Math.PI * 2);    "righthip": Unknown word.
  //left hip
  canvasCtx.arc(lefthip[i].x, lefthip[i].y, 8, 0, Math.PI * 2);    "lefthip": Unknown word.

  canvasCtx.fillStyle = "#AAFF00";
  canvasCtx.fill();
} ← #184-202 for (let i = 0; i < 3; i++)

if (
  inRangeLeftHand == true &&
  inRangeRightHand == true &&
  inRangeRightHip == true &&
  inRangeLeftHip == true
) {
  if (dir == 0) {
    count = count + 1;
    speak(count);
    dir = 1;
    console.log(count);
  } ← #210-215 if (dir == 0)
} ← #209-216 { if (dir == 0) { count = count + 1; speak(count); dir = 1; ...

if (
  !(
    inRangeLeftHand == true &&
    inRangeRightHand == true &&
    inRangeRightHip == true &&
    inRangeLeftHip == true
  ) ← #219-224 !
) {
  dir = 0;
}

canvasCtx.font = "30px aerial";
canvasCtx.fillText(leftHandAngle, leftHand[1].x + 20, leftHand[1].y + 20);
canvasCtx.fillText(
  rightHandAngle,
  rightHand[1].x - 120,
```