Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ ДЛЯ РОЗПІЗНАВАННЯ ТА КОНТРОЛЮВАННЯ ПРАВИЛЬНОСТІ ВИКОНАННЯ ФІЗИЧНИХ ВПРАВ У РЕАЛЬНОМУ ЧАСІ
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-21-1
Голубович Є. Д.
(прізвище, ініціали)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Інформатика
(повна назва освітньої програми)
Керівник доц. Творошенко І. С.
(посада, прізвище, ініціали)

Допускається до захисту

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

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Голубовичу Євгенію Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення вебзастосунку для розпізнавання та контролювання правильності виконання фізичних вправ у реальному часі

затверджена наказом університету від 19 травня 2025 року № 381Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 25 травня 2025 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, мобільні застосунки, бібліотека для машинного навчання та комп'ютерного зору TensorFlow.js, бібліотека для створення інтерфейсів користувача React, бібліотека для управління станом застосунку Redux, утилітарний набір інструментів Tailwind CSS, програмне забезпечення для моделювання систем Rational Rose.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз сучасних вебзастосунків для розпізнавання та контролювання правильності виконання фізичних вправ в Україні та за кордоном.
2. Особливості розроблення вебзастосунку для розпізнавання та контролювання правильності виконання фізичних вправ у реальному часі.
3. Етапи розроблення вебзастосунку для розпізнавання та контролювання правильності виконання фізичних вправ у реальному часі.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми, мета роботи, постановка задачі, етапи виконання роботи, перспективи подальшої роботи, зображення графічних інтерфейсів вебзастосунків, діаграма варіантів використання, діаграма класів, зображення частин коду застосунку.

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

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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	07.04.2025	
2	Аналіз завдання, підбір літератури	08.04.25-13.04.25	
3	Аналіз літератури з досліджуваної проблеми	14.04.25-16.04.25	
4	Аналіз сучасних методів розпізнавання	17.04.25-24.04.25	
5	Моделювання вебзастосунку	25.04.25-29.04.25	
6	Програмна реалізація	30.04.25-11.05.25	
7	Оформлення пояснювальної записки	12.05.25-20.05.25	
8	Перевірка на нормоконтроль	21.05.25-01.06.25	
9	Перевірка на плагіат	21.05.25-01.06.25	
10	Рецензування	21.05.25-01.06.25	
11	Підготовка презентації та доповіді	21.05.25-18.06.25	
12	Занесення роботи в електронний архів	02.06.25-18.06.25	
13	Попередній захист кваліфікаційної роботи	02.06.25-18.06.25	

Дата видачі завдання 7 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ _____ доц. Творошенко І. С.____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 93 с., 1 табл., 89 рис., 51 джерело.

ВЕБЗАСТОСУНОК, КОМП'ЮТЕРНИЙ ЗІР, РОЗПІЗНАВАННЯ ФІЗИЧНИХ ВПРАВ, МАШИННЕ НАВЧАННЯ, КЛЮЧОВІ ТОЧКИ, СИСТЕМА ТРЕНУВАНЬ, ОБРОБКА ДАНИХ У РЕАЛЬНОМУ ЧАСІ.

Об'єктом роботи є процес розроблення вебзастосунку для розпізнавання та контролювання правильності виконання фізичних вправ у реальному часі.

Метою роботи є вебзастосунок, реалізований засобами комп'ютерного зору для особистих тренувань.

Використано методи комп'ютерного зору для перетворення даних у модель з ключовими точками, яку можна аналізувати для досягнення контролю правильності виконання фізичних вправ. Проведено детальний аналіз сучасних методів розпізнавання. Виконано функціональне та структурне моделювання, здійснено опис можливостей вебзастосунку.

У результаті роботи розроблено вебзастосунок для покращення фізичної форми людини.

WEB APPLICATION, COMPUTER VISION, RECOGNITION OF PHYSICAL EXERCISES, MACHINE LEARNING, KEYPOINTS, TRAINING SYSTEM, DATA PROCESSING IN REAL TIME.

The object of the work is the process of developing a web application for recognizing and monitoring the correctness of physical exercises in real time.

The aim of the work is a web application implemented using computer vision tools for personal training.

Computer vision methods were used to convert the data into a key point model that can be analyzed to achieve the correctness of physical exercise control. Detailed research on modern recognition methods was carried out. Functional and structural modeling was performed, and the capabilities of the web application were described.

As a result, a web application was developed to improve physical form of a person.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 7
Вступ 9
1 Аналіз існуючих вебзастосунків для розпізнавання та контролювання правильності виконання фізичних вправ 10
1.1 Аналіз сучасних вебзастосунків для розпізнавання та контролювання правильності виконання фізичних вправ в Україні та за кордоном 10
1.2 Класифікація фізичних вправ 17
1.3 Основні вимоги до правильності виконання фізичних вправ 18
1.4 Аналіз літературних джерел щодо апробації результатів розроблення вебзастосунків для розпізнавання та контролювання правильності виконання фізичних вправ у реальному часі 21
1.5 Постановка задачі 26
2 Особливості розроблення вебзастосунку для розпізнавання та контролювання правильності виконання фізичних вправ у реальному часі .. 27
2.1 Детальний аналіз сучасних методів розпізнавання 27
2.2 Особливості методів контролю правильності виконання фізичних вправ у реальному часі 34
2.3 Формулювання методики розпізнавання та контролювання правильності виконання фізичних вправ у реальному часі 37
2.4 Моделювання структури вебзастосунку для розпізнавання та контролювання правильності виконання фізичних вправ у реальному часі……..44
3 Етапи розроблення вебзастосунку для розпізнавання та контролювання правильності виконання фізичних вправ у реальному часі 50
3.1 Вибір інструментальних засобів реалізації вебзастосунку для розпізнавання та контролювання правильності виконання фізичних вправ у реальному часі 50

3.2 Етапи програмної реалізації розпізнавання та контролювання правильності виконання фізичних вправ у реальному часі 52
3.3 Тестування розробленого вебзастосунку та аналіз результатів 76
3.4 Перспективи подальшої розробки 85
Висновки 87
Перелік джерел посилання 88

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

МН – машинне навчання

Вебзастосунок – застосунок, запущений у браузері (клієнт), а сервером є вебсервер

ШІ – штучний інтелект

OpenCV – Open Computer Vision Library (бібліотека для роботи з обличчям)

CNN – Convolutional Neural Networks (згорткові нейронні мережі, використовуються для обробки зображень у комп’ютерному зорі)

HPE – Human Pose Estimation (технологія комп’ютерного зору, яка визначає положення ключових точок людського тіла на зображеннях або відео)

GUI – Graphical User Interface (графічний інтерфейс користувача, що забезпечує зручне налаштування та управління системою)

YOLO – You Only Look Once (архітектура CNN, яка використовується для розпізнавання множини об’єктів на зображенні)

SSD – Single Shot Detector (за принципом схожа на YOLO, але в мережі для вилучення ознак використовується 16 шарів)

FPN – Feature Pyramid Networks (різновид мережі типу SSD)

CV – Computer Vision (комп’ютерний зір)

HOG – Histogram of Oriented Gradients (гістограма орієнтованих градієнтів)

SIFT – Scale-Invariant Feature Transform (масштабно-інваріантне перетворення ознак)

SURF – Speeded-Up Robust Features (пришвидшені надійні функції)

FAST – Features from Accelerated Segment Test (функції тестування прискореного сегмента)

BRIEF – Binary Robust Independent Elementary Features (двійкові надійні незалежні елементарні функції)

ORB – Oriented FAST and Rotated BRIEF (орієнтований FAST і обернутий BRIEF)

SVM – Support Vector Machine (метод опорних векторів)

CPU – Central Processing Unit (центральний процесор)

GPU – Graphics Processing Unit (графічний процесор)

UML – Unified Modeling Language (уніфікована мова моделювання)

CSS – Cascading Style Sheets (каскадні таблиці стилів)

HTML – Hypertext Markup Language (мова розмітки гіпертексту)

DOM – Document Object Model (об’єктна модель документа)

WebGL – Web Graphics Library (графічна бібліотека для вебзастосунків)

ВСТУП

Зростання інтересу сучасного суспільства до здорового способу життя призвело до розвитку технологій, що підтримують фізичну активність. Одним із ключових аспектів ефективного тренування є правильний вибір режиму занять, від якого безпосередньо залежить здоров’я людини та досягнення нею бажаних результатів. Традиційно цим займалися персональні тренери, але сучасні цифрові рішення дозволяють значно спростити процес контролю та вдосконалення техніки виконання вправ, та зробити його доступним для широкої аудиторії. Це особливо актуально в умовах зростання попиту на дистанційні тренування та індивідуальні програми навчання.

Розвиток комп’ютерного зору та методів машинного навчання відкриває нові можливості для автоматичного контролю правильності виконання вправ у режимі реального часу. Вебзастосунки, які використовують ці технології, можуть розпізнавати рухи користувача та надавати зворотний зв’язок щодо правильності виконання вправи.

Актуальність роботи полягає у необхідності розробки доступного і зручного інструменту для персональних тренувань, що застосовує та поєднує новітні технології для розширеного контролю фізичної активності. Вебзастосунок, розроблення якого описано у цій роботі, покликаний вирішити це завдання, використовуючи методи комп’ютерного зору для аналізу ключових точок тіла під час виконання вправ. Це дозволить користувачам самостійно контролювати якість своїх тренувань в режимі реального часу без постійної присутності тренера.

1 АНАЛІЗ ІСНУЮЧИХ ВЕБЗАСТОСУНКІВ ДЛЯ РОЗПІЗНАВАННЯ ТА КОНТРОЛЮВАННЯ ПРАВИЛЬНОСТІ ВИКОНАННЯ ФІЗИЧНИХ ВПРАВ

1.1 Аналіз сучасних вебзастосунків для розпізнавання та контролювання правильності виконання фізичних вправ в Україні та за кордоном

У сфері фізичних вправ є багато застосунків, які дозволяють складати плани тренувань і коригувати їх у залежності від фізичного стану людини. Здебільшого, вони спрямовані на мобільні платформи.

На рисунках 1.1 та 1.2 показано інтерфейс мобільного застосунку Home Workout [1].

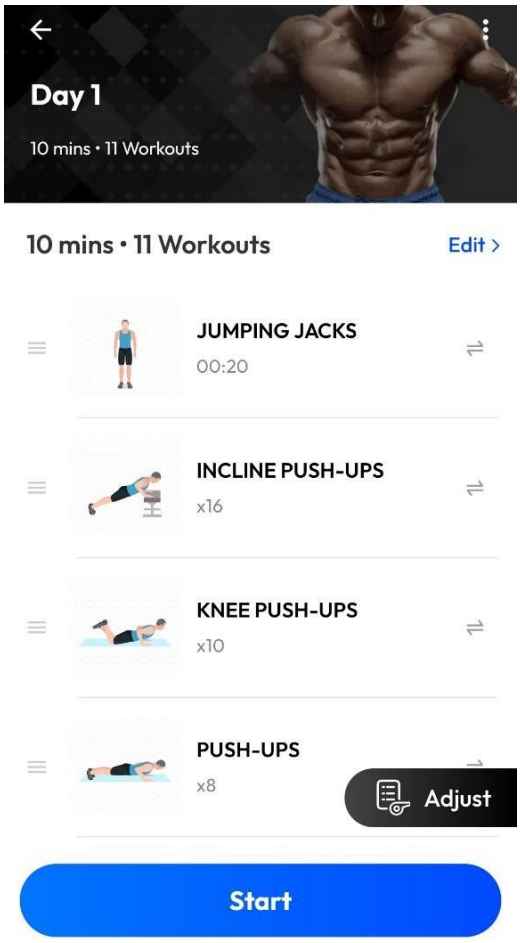


Рисунок 1.1 – Скриншот персональної програми тренувань в Home Workout

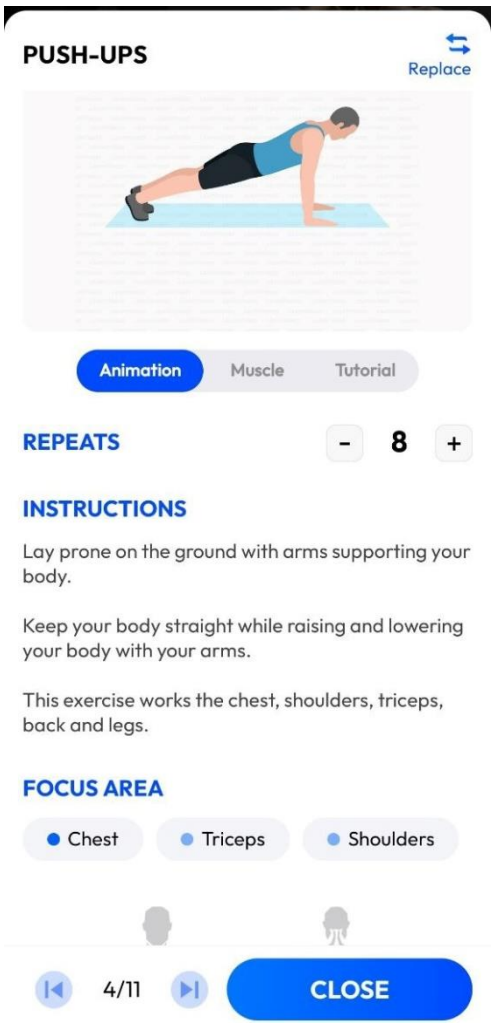


Рисунок 1.2 – Скриншот інструкції з виконання фізичної вправи у застосунку Home Workout

Переваги мобільного застосунку Home Workout:

– наявність великої кількості планів тренувань для різних цілей;

– присутня можливість тренуватися зі спортивним обладнанням;

– можливість заміни вправ, кількості повторень чи зміни рівню складності тренувань.

Недоліки мобільного застосунку Home Workout:

– застосунок не виконує перевірку виконання вправ;

– відсутність рекомендацій щодо вибору програм тренувань;

– певні функції є доступними лише при наявності платної версії;

– максимальна кількість підходів на вправу обмежена одним за одне тренування.

FitAI – мобільний застосунок, що надає можливість обирати згенеровані за допомогою ШІ програми тренувань (рис. 1.3) [2].

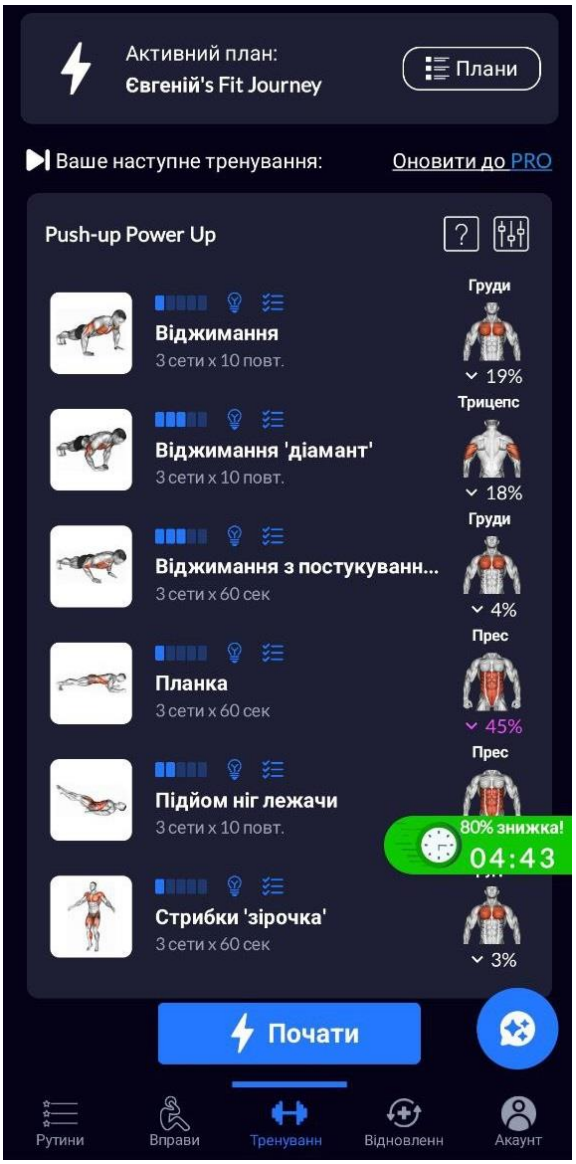


Рисунок 1.3 – Скриншот програми тренувань з FitAI

Переваги мобільного застосунку FitAI:

– вправи поділено за рівнями та прописано детальні інструкції до них (рис. 1.4);

– інтеграція ChatGPT у застосунку;

– можливість запису вимірювань для подальшої корекції тренувань;

– можливість тренуватися з різноманітним інвентарем;

– можливість генерації планів тренувань за допомогою ШІ (рис. 1.3).

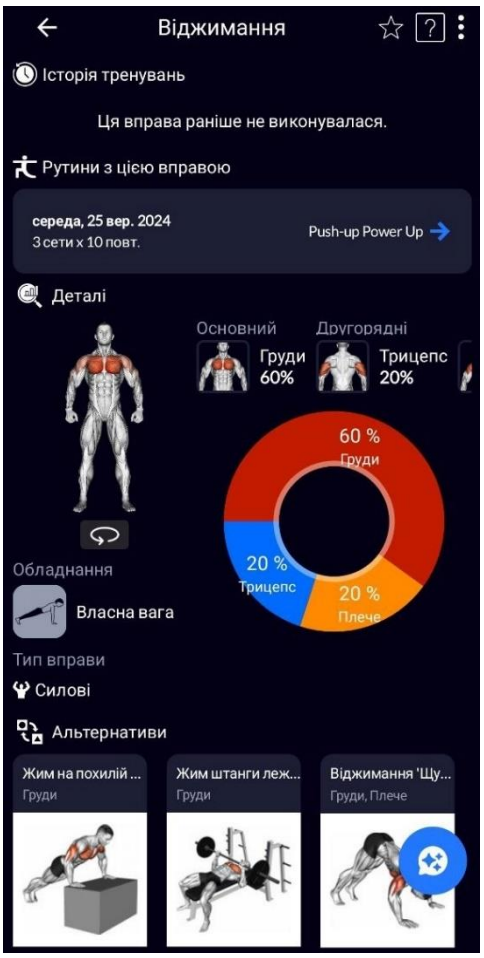


Рисунок 1.4 – Скриншот інструкції з виконання фізичної вправи в FitAI

Недоліки мобільного застосунку FitAI:

– велика кількість функціоналу доступна лише за умови придбання підписки;

– застосунок не виконує перевірку виконання вправ;

– більшість програм базується на тренуванні окремих груп м’язів з великим обсягом навантажень, що не завжди є оптимальним для досягнення поставлених цілей.

Personal Fitness Trainer AI – вебзастосунок, що використовується для розпізнавання виконання людиною фізичних вправ у реальному часі з використанням OpenCV (рис. 1.5) [3]. У ньому відсутня можливість вибору чи створення плану, графічний інтерфейс обмежений демонстрацією камери користувача з показом кількості виконаних вправ та контрольних точок на тілі людини.

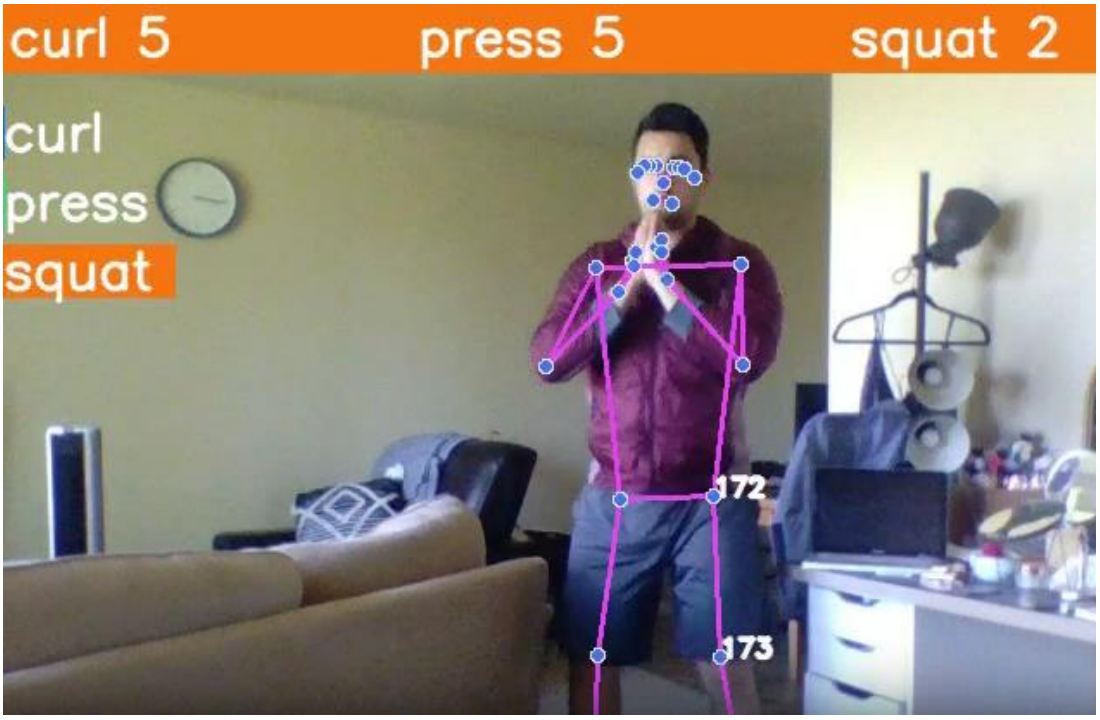


Рисунок 1.5 – Скриншот роботи вебзастосунку Personal Fitness Trainer AI

Переваги вебзастосунку Personal Fitness Trainer AI:

– наявність перевірки виконання вправ у режимі реального часі з автоматичним підрахунком кількості виконаних вправ;

– велика точність розпізнавання вправ;

– відображення моделі контрольних точок поверх тіла людини.

Недоліки вебзастосунку Personal Fitness Trainer AI:

– кількість вправ обмежена трьома;

– застосунок не має планів тренувань;

– відсутня опція вимкнення показу моделі контрольних точок;

– програма не надає рекомендацій щодо виконання вправ та не містить інструкцій.

FormCheck – мобільний застосунок, що використовується для відслідковування різниці у формі виконання людиною повторень під час підходу [4].

Переваги мобільного застосунку FormCheck:

– дозволяє створювати власні шаблони для відслідковування форми людини під час виконання вправ;

– фіксує кути нахилу між контрольними частинами тіла та формує звітність про вправу (рис. 1.6);

– підраховує кількість повторень та записує моменти недотримання техніки виконання.

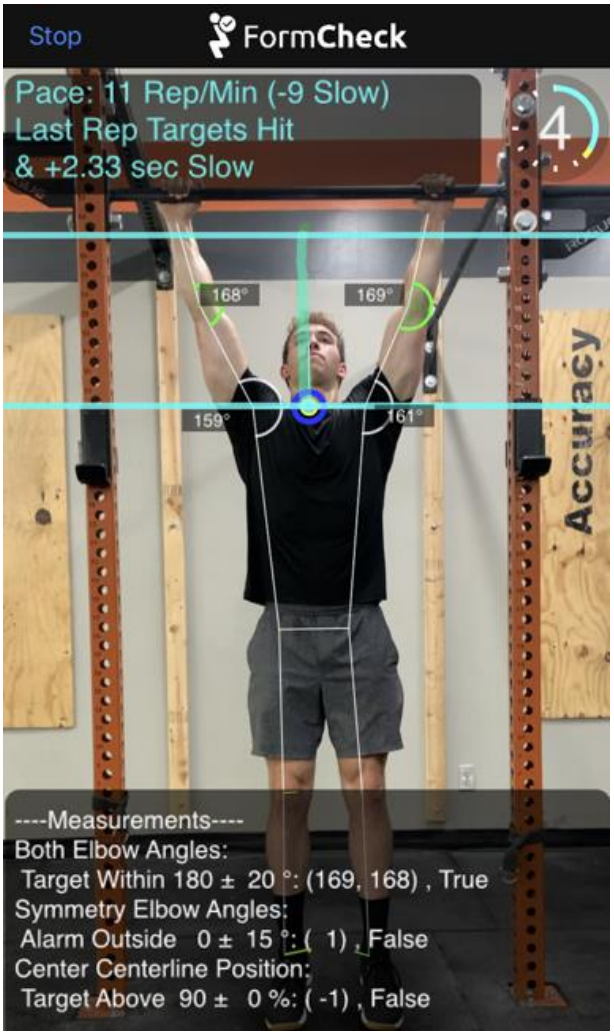


Рисунок 1.6 – Скриншот роботи застосунку FormCheck

Недоліки мобільного застосунку FormCheck:

– відсутність можливості перевірки форми виконання у реальному часі, дозволяється працювати лише з готовими відео;

– відсутні плани тренувань та дуже мала кількість готових шаблонів;

– головний функціонал застосунку доступний лише за умови придбання підписки;

– інтерфейс є досить складним для звичайних користувачів.

Agit – мобільний застосунок для запису тренувань та розпізнавання виконання фізичних вправ за допомогою методів комп’ютерного зору, дозволяє виконувати та створювати плани тренувань (рис 1.7) [5].

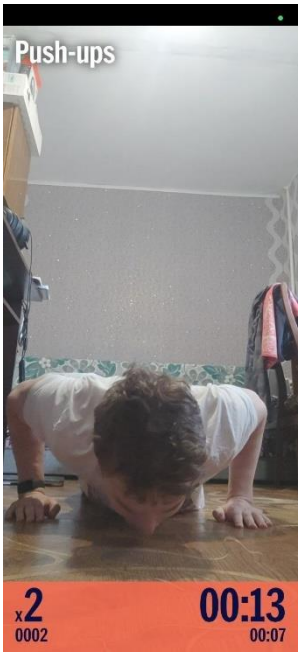


Рисунок 1.7 – Скриншот з виконання тренування у застосунку Agit

Переваги мобільного застосунку Agit:

– дає можливість створювати плани тренувань за часом чи за кількістю повторень;

– відслідковує виконання вправ у реальному часі;

– є можливість змагатися з іншими користувачами.

Недоліки мобільного застосунку Agit:

– відсутність можливості перегляду моделі контрольних точок;

– відсутність рекомендацій щодо виконання вправ у реальному часі;

– аналітика з тренувань є досить обмеженою;

– застосунок не має системи тренувань для подальшого прогресу при підвищенні фізичних показників людини.

Отже, існуючі застосунки не використовують усі можливості комп’ютерного зору у поєднанні з методиками для оптимальних і зручних персональних тренувань.

1.2 Класифікація фізичних вправ

Фізичні вправи класифікуються за різними критеріями, залежно від мети тренувань, характеру рухової активності та рівня впливу на організм. Основні класифікації фізичних вправ включають такі:

− за метою виконання;

− за типом рухової активності;

− за рівнем впливу на організм;

− за руховими якостями;

− за використанням обладнання;

− за рівнем інтенсивності.

За метою виконання вправи поділяються на:

− загальнорозвиваючі (спрямовані на зміцнення всіх груп м’язів і підтримку загальної фізичної форми);

− спеціальні (використовуються для розвитку конкретних фізичних якостей або навичок);

− реабілітаційні (спрямовані на відновлення після травм або хвороб).

За типом рухової активності маємо:

− аеробні (вправи, які виконуються тривалий час при помірній інтенсивності, з акцентом на витривалість серця і легені);

− анаеробні (вправи, що спрямовані на розвиток сили та потужності м’язів, як правило, виконуються в короткі інтервали при високій інтенсивності).

За рівнем впливу на організм поділяються на:

− загальні (впливають на все тіло і включають декілька груп м’язів);

− локальні (націлені на конкретні групи м’язів або частини тіла).

За руховими якостями маємо:

− силові (розвивають силу м’язів);

− швидкісні (розвивають швидкість руху і реакції);

− координаційні (спрямовані на розвиток рівноваги і точності рухів);

− гнучкість (вправи для покращення амплітуди рухів в суглобах).

За використанням обладнання поділяються на:

− вправи з вагою тіла;

− вправи зі спортивним інвентарем.

За рівнем інтенсивності:

− низькоінтенсивні (вправи, що виконуються на низькому рівні напруження);

− середньоінтенсивні (помірна активність, що підвищує частоту серцевих скорочень);

− високоінтенсивні (включають швидкі, вибухові рухи).

Отже, для підтримки фізичної форми в домашніх умовах звичайній людині найкраще підходять вправи, які не потребують спеціального обладнання та дозволяють опрацьовувати різні групи м’язів. Бажано задіяти максимальну кількість груп м’язів при мінімальній кількості вправ. Це мають бути загальнорозвиваючі, анаеробні, загальні, силові вправи. Для досягнення виконання цих умов оптимальними будуть такі базові вправи як віджимання, присідання та скручування на прес. Такі тренування підходять для людей будь-якого рівня підготовки і допомагають підтримувати загальний тонус та здоров’я тіла.

1.3 Основні вимоги до правильності виконання фізичних вправ

Правильність виконання фізичних вправ є ключовим фактором для запобігання травмам і досягнення максимальної ефективності. Розглянемо основні вимоги до техніки виконання віджимань, присідань та скручування на прес.

Віджимання є базовою вправою для розвитку м’язів грудей, рук, плечей і тулуба.

Вимоги до виконання:

− початкова позиція:

1) тіло займає пряму лінію від голови до п’ят. Долоні розташовані на ширині плечей або трохи ширше;

2) пальці ніг знаходяться на підлозі, ноги разом або трохи розведені;

3) руки повинні бути під плечима, пальці рук дивляться вперед;

− техніка руху:

1) опускати тіло, згинаючи лікті під кутом 45 градусів до тулуба;

2) грудні м’язи мають опуститися якомога ближче до підлоги, але не торкатися її;

3) лікті не повинні виходити надто широко в сторони, щоб уникнути навантаження на плечі;

4) тіло залишається в одній лінії (поперек не прогинається, сідниці не піднімаються вгору);

− дихання:

1) вдих при опусканні;

2) видих при підйомі;

− типові помилки:

1) неправильна постановка рук (занадто вузько або широко);

2) прогин попереку або підняття тазу вгору;

3) недостатня амплітуда руху.

Присідання є однією з найефективніших вправ для розвитку нижніх м’язів тіла (стегна, сідниці, квадрицепси) та м’язів кора.

Вимоги до виконання:

− початкова позиція:

1) ноги на ширині плечей або трохи ширше, стопи розвернуті назовні приблизно на 15-30 градусів;

2) спина повинна бути прямою, плечі злегка опущені назад, голова дивиться прямо перед собою;

− техніка руху:

1) починати рух з відведення тазу назад, ніби намагаємось сісти на невидимий стілець;

2) коліна повинні рухатися у напрямку до носків, але не виходити за їх межі;

3) стегна опускаються до рівня паралелі з підлогою або нижче, залежно від гнучкості людини;

4) вага тіла повинна бути на п’ятах, а не на носках;

− дихання:

1) вдих при опусканні;

2) видих при підйомі;

− типові помилки:

1) округлення спини під час виконання вправи;

2) занадто сильний нахил тулуба вперед;

3) перенесення ваги на носки, що може спричинити перенавантаження колін.

Скручування є базовою вправою для розвитку прямих м’язів живота та м’язів кора.

Вимоги до виконання:

− початкова позиція:

1) лягти на спину, коліна зігнути, стопи стоять на підлозі;

2) руки можна тримати за головою (без натискання) або схрестити на грудях;

− техніка руху:

1) підіймати верхню частину тулуба, напружуючи м’язи преса. Рух виконується завдяки напруженню м’язів живота, а не за рахунок рук або шиї;

2) поперек залишається на підлозі, піднімається лише верхня частина тулуба;

3) під час підйому тримати шию в нейтральному положенні, щоб уникнути навантаження на неї;

− дихання:

1) видих під час підйому;

2) вдих при опусканні у вихідне положення;

− типові помилки:

1) підтягування голови за руками, що може спричинити травми шиї;

2) виконання руху за рахунок імпульсу, а не м’язової напруги;

3) надмірна амплітуда (підйом всього тулуба замість часткового скручування).

Отже, дотримання правильної техніки виконання вправ є важливим для ефективності тренувань і запобігання травмам. Правильне дихання, контроль за положенням тіла та м’язовою напругою допомагають досягти найкращих результатів і зберегти здоров’я.

1.4 Аналіз літературних джерел щодо апробації результатів розроблення вебзастосунків для розпізнавання та контролювання правильності виконання фізичних вправ у реальному часі

У роботі [6] автор описує проблему реалізації моделі, яка буде спроможна вирішувати актуальну проблему захвату рухів кінцівок людини за вебкамери використовуючи методи глибокого навчання. Технології з визначення людських ключових точок (Human Pose Estimation, HPE) є фундаментальною областю досліджень у галузі комп’ютерного зору, яка швидко розвивається. Вона спрямована на оцінку положення та орієнтації суглобів людського тіла на зображеннях або відео, що дозволяє комп’ютеру розуміти й аналізувати пози людського тіла. Автор виявив, що поява глибокого навчання, зокрема згорткових нейронних мереж (Convolutional Neural Networks, CNN), значно покращила сучасний рівень HPE, завдяки таким моделям, як Stacked Hourglass Networks, Simple Baselines і Vision

Transformers, які демонструють надзвичайну продуктивність як з точки зору точності, так і надійності.

У роботі [7] автором розглянуто загальний підхід до розпізнавання окремих частин людського тіла на відеозображеннях, які надходять з камер на різноманітних пристроях з обмеженими обчислювальними ресурсами. Зокрема, досліджується розпізнавання та аналіз жестів руки для розробки безконтактного графічного інтерфейсу (Graphical User Interface, GUI). Особливістю зазначеної задачі є вимога до можливості виконання їх програмної реалізації на малопотужних пристроях, які можуть працювати автономно без використання додаткових серверних ресурсів. Для задачі побудовано модель, в основі якої лежить множина, що складається з малої кількості ключових точок. За рахунок цього вдалося розробити швидкі та високоефективні алгоритми детекції та аналізу. Програмна реалізація алгоритмів задачі виконана з використанням нейронних мереж бібліотеки MediaPipe для операцій детекції та скелетонізації і SciKit для безпосереднього розпізнавання жестів. Завдяки малій кількості використаних ключових параметрів кисті руки на завершальному етапі виявилося ефективним використання простої повнозв’язної мережі.

У статті [8] проаналізовано існуючі системи розпізнавання людей та дано їх характеристику. Окремо описано особливості двоступеневого та одноступеневого розпізнавання образів, наведено методики підвищення продуктивності глибоких нейронних мереж за допомогою зменшення кількості операцій. Подано особливості роботи нейронних мереж на основі YOLOv3 та охарактеризовано архітектуру досліджуваного алгоритму. Описано програмну реалізацію для YOLOv3 з участю фреймворка Darknetta бібліотеки Tensorflow. Описано процес тренування досліджуваної нейтронної мережі. Отримані результати навчання свідчать, що покращена мережа має хорошу продуктивність в задачах розпізнавання і класифікації цілей.

У дипломному проєкті [9] автором розглянуто проблему розробки та реалізації програмного модуля, який буде розпізнавати позу людини.

Проаналізувавши предметну область, існуючі системи розпізнавання пози людини, було розроблено структуру модуля та алгоритмічне забезпечення. Проаналізувавши інструментарій та мови програмування, для розробки програмного продукту було обрано мову програмування Python, відкриту нейромережну бібліотеку Keras для побудови, тренування та використання моделей глибокого навчання. У процесі розробки дипломного проєкту було вивчено та порівняно декілька методів розпізнавання пози. Також проведено тестування модуля наступними способами: тестування на реальних зображеннях, де користувач може підготувати зображення, які містять людей з різними позами, та тестування з використанням даних з розміченою позою, де завантажується зображеннях з публічних наборів даних, для якого була вже розмічена поза і порівнюються з отриманими даними.

У роботі [10] запропонований підхід до розробки телемедичної системи з використанням технологій комп’ютерного зору. Основною метою розробки є нагляд та ведення хворих під час відновлення рухової діяльності дистанційно в умовах дому. Але програмний додаток може бути використаний у спортіндустрії під час дистанційних тренувань.

У роботі [11] розглянуто методи, що використовуються для відстеження тіла людини у просторі. Кожен метод був детально описаний з наведенням всіх формальних моделей та зв’язків. Також кожен метод несе за собою окреме призначення та буде використовуватися у різних задачах. Було показано, що у відповідності до поставленого завдання для вирішення, необхідно обирати конкретні методи, які поділяються на ті, що використовують різноманітні додаткові датчики та пристрої, та ті, яким достатньо лише однієї камери. Отримано дані, що методи відстеження тіла за допомогою додаткових датчиків, надають найбільш точні результати. Проте також було виявлено під час виконання роботи, що і з використанням методів зчитування тіла з зображення шляхом використання методів комп’ютерного зору можна досягти достатньо великої точності.

У роботі [12] автором було створено та оптимізовано комп’ютерну систему для ефективної оптимізації фізичних навантажень спортсменів. Впровадження комп’ютерної системи для оптимізації фізичних навантажень дозволяє аналізувати та коригувати навантаження з урахуванням індивідуальних потреб і можливостей спортсмена, покращуючи їхні результати та знижуючи ризик травм Було проведено порівняльний аналіз існуючих систем-конкурентів. Виходячи з результатів цього аналізу, було розроблено комп’ютерну систему для оптимізації фізичних навантажень спортсменів. Реалізована система дозволяє тренерам і спортсменам аналізувати навантаження, планувати тренування та здійснювати коригування в режимі реального часу, що допомагає спортсменам досягати кращих результатів.

У роботі [13] описуються особливості популярних архітектур CNN: R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN, YOLO, SSD, Feature Pyramid Network (FPN), RetinaNet. Розглянуто складові елементи та принципи роботи цих архітектур. Проаналізовано продуктивність архітектури CNN та співвідношення точності розпізнавання до часу висновку. Архітектуру YOLOv3 було обрано, тому що вона має хорошу точність до співвідношення часу висновку, обробляє зображення швидше, ніж інші архітектури CNN, добре працює в обробці зображень у реальному часі, а передбачення (розташування об’єктів і класи) створюються з однієї мережі. Методи пропозиції регіону обмежують класифікатор певним регіоном. YOLO обробляє все зображення в режимі прогнозування меж. У додатковому 81 контексті YOLO демонструє менше хибних спрацьовувань у фонових регіонах, а також застосовує просторову різноманітність у прогнозах.

У ході виконання роботи [14] було розглянуто питання пов’язані з вибором типу нейронної мережі, яка краще справляється з завданням розпізнавання об’єктів на зображені. Наведено характеристику предметного середовища. Детально розглянуто алгоритми навчання мережі та методики для покращення точності прогнозування. Описано вимоги та технології для

створення даної системи. Для розробки системи розпізнавання об’єктів на зображенні використано мову програмування Python, середовищем для розробки обрано Jupyter Notebook. Також використано бібліотеку Tensorflow для навчання нейронної мережі та підготовки тренувального набору. Дослідивши процес навчання моделі, можна з впевненістю сказати, що точність розпізнавання моделі залежить від кількості епох, тобто проходу по всіх елементах тренувального набору. В кінці навчання, після завершення двадцятої епохи, точність на перевірочних даних становила 99%. Досліджено, що збільшуючи кількість епох, збільшується якість нейронної мережі, але це відбувалося тільки до п’ятої епохи, наступні дії не давали великого приросту точності мережі.

У роботі [15] було розглянуто методи обробки зображень на базі згорткових нейронних мереж, які відповідають поставленим вимогам та вирішують актуальну проблему за допомогою методів машинного навчання застосовуючи комп’ютерний зір забезпечуючи велику точність та гнучкість програмних засобах у модифікації складових частин. Для вирішення поставленої задачі були використані два компонента з CNN та Transformer архітектур, щоб провести демонстрацію поєднання двох типів моделей для досягнення кращих результатів в порівнянні зі звичайними згортковими мережами. Основна мова програмування та обробки даних − Python. Щоб використовувати моделі машинного застосувався фреймворк Tenserflow, Keras та Airflow. Гнучкість підібраних інструментів та підібраної архітектури дозволить вносити зміни без кардинальних втрат часу та ресурсів. Під час досліджень задля вирішення задачі комп’ютерного зору були зроблені певні висновки:

− від якісних даних залежить левова частка результату у розпізнанні образів;

− від обраної архітектури та її компонентів залежить ефективність моделі машинного навчання. У весь час потрібно балансувати між витратами ресурсів та точністю результатів;

− потрібно проводити тестування програмних засобів, щоб якість та функціональність програмного забезпечення відповідала вимогам користувача.

Проаналізувавши літературні джерела [6–15], можна зробити висновок, що використання згорткових нейронних мереж дозволяє отримати високу точність класифікації вхідних даних при відстеженні статичних поз людини та її рухів. У спорті CNN ще не набули достатнього поширення і на ринку бракує застосунків у відкритому просторі для розпізнавання та правильного виконання фізичних вправ у реальному часі, тому у цій сфері присутня необхідність впровадження нових рішень.

1.5 Постановка задачі

Таким чином, розробка інтерактивного сайту з можливістю відстеження техніки виконання рухів людини та прогресу результату тренувань є актуальним завданням.

Об’єктом роботи є процес розроблення вебзастосунку для розпізнавання та контролювання правильності виконання фізичних вправ у реальному часі.

Метою роботи є вебзастосунок, реалізований засобами комп’ютерного зору для особистих тренувань.

Можемо визначити завдання для досягнення поставленої мети:

− проаналізувати сучасні програми та сайти спортивного напрямку з можливістю контролювання якості виконання фізичних вправ;

− провести аналіз існуючих методів розпізнавання на базі CNN, порівняти та обрати оптимальний;

− розробити повноцінний вебзастосунок на базі обраного методу.

2 ОСОБЛИВОСТІ РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ ДЛЯ РОЗПІЗНАВАННЯ ТА КОНТРОЛЮВАННЯ ПРАВИЛЬНОСТІ ВИКОНАННЯ ФІЗИЧНИХ ВПРАВ У РЕАЛЬНОМУ ЧАСІ

2.1 Детальний аналіз сучасних методів розпізнавання

Задача розпізнавання об’єктів на відео належить до сфери комп’ютерного зору, оскільки потребує аналізу та інтерпретації візуальної інформації. CV поділяє задачі на такі як розпізнавання, відстеження у русі, відновлення сцени та відновлення зображень чи відео [16]. Поставлене завдання відноситься відразу до розпізнавання та відстеження у русі. Оскільки обробка відео відбувається шляхом почергової обробки кожного кадру, то першочергово треба розглядати існуючі шляхи детекції об’єктів.

Для визначення наявності об’єктів на зображеннях застосовують інженерію ознак. Вона відповідає за процес обчислення важливих ознак, на основі яких можливо буде проводити подальшу обробку даних. Найпоширенішими методами є HOG, SIFT, SURF, FAST, ORB, BRIEF [17].

Розглянемо переваги й недоліки кожного окремо.

Переваги HOG: добре працює в задачах, де потрібно виявити форми.

Недоліки HOG: не є інваріантним до масштабу та обертання.

Переваги SIFT: потужний метод для виділення інваріантних до масштабу та обертання ознак.

Недоліки SIFT:

− має фокус на виявленні точок у другорядних місцях на зображеннях;

− не забезпечує прямого рішення для відстеження поз;

− патентні обмеження.

Переваги SURF: виявляє важливі ознаки, інваріантні до масштабу та обертання.

Недоліки SURF:

− не орієнтований на групи точок;

− патентні обмеження.

Переваги FAST: швидкий метод знаходження ключових точок.

Недоліки FAST:

− менш точний через відсутність можливості ефективної роботи з масштабами та орієнтаціями;

− не надає дескрипторів для точок, що важливо для відстеження руху.

Переваги ORB:

− баланс між точністю та швидкістю;

− ефективно працює з масштабом та обертальною інваріантністю.

Недоліки ORB:

− не дає структурної інформації про всю форму об’єкта;

− менша точність для складних ситуацій для HPE.

Переваги BRIEF: дуже швидкий для створення бінарних дескрипторів.

Недоліки BRIEF:

− не враховує обертання та масштабування;

− низька точність для визначення ключових точок людського тіла.

У результаті для HPE віддають перевагу використанню HOG для виявлення частин тіла, бо він надає чіткий, структурований опис контурів і форм, швидкий у реалізації, легко поєднуваний з класичними моделями машинного навчання. Найпоширенішим способом проведення оцінки поз людини до появи сучасних методів глибокого навчання були Pictorial Structures на базі градієнтів та розширення Deformable Part Models, HOG + SVM, графові моделі, статистичні методи. Розглянемо окремо кожен спосіб.

Pictorial Structures – спосіб представлення людини у якості набору з’єднаних частин тіла [18]. Модель працює за принципом пошуку найбільш ймовірного розташування кожної частини тіла, враховуючи локальні ознаки, довжини, кути. Зазвичай у моделі застосовуються прості ознаки або локальні шаблони. Детектор застосовують для кожної частини тіла. Просторові зв’язки прийнято будувати на основі гаусівських моделей, які визначають допустимі деформації, кути та відстані.

Ціль – знайти конфігурацію з мінімальними загальними витратами або максимальній правдоподібності на основі байєсівській моделі. Формально можна представити як:

$$P(L|I) \propto P(I|L)P(L), \tag{2.1}$$

де L – конфігурація частин;

I – зображення;

$P(I|L)$ – наскільки добре кожна частина вписується у відповідному місці;

$P(L)$ – наскільки правдоподібна така конфігурація частин.

Переваги такого способу:

− інтерпретованість;

− врахування анатомії;

− можливість праці з частковими спостереженнями.

Недоліки:

− висока чутливість до шуму;

− нестабільна точність при варіативності поз;

− відсутність можливості автоматичного навчання;

− повільна.

Deformable Part Models (DPM) – являє собою розширення ідеї Pictorial Structures [19]. Її нововведенням стала можливість відстеження гнучких частин тіла, що дозволило покращити точність врахування поз. Модель здатна виділяти частини, що обертаються чи масштабуються. У якості ознак можна застосовувати HOG чи SIFT. Навчання перейшло на розширення SVM, що дало змогу автоматично виявляти положення частин тіла без явної розмітки. На виході формується декілька варіантів в межах моделі. У якості формули обчислення деформації виступає:

$$d(x, y) = ax^2 + by^2 + c|x| + d|y|. \tag{2.2}$$

Як можна помітити, накладається квадратичний та лінійний штрафи. Гаусівський (квадратичний) штраф залишився ще з Pictorial Structures, а от лінійний штраф став нововведенням, що додало гнучкості до моделі. Зазнала змін і загальна формула оцінки:

$$score = RootScore + \sum_i (PartScore_i - DeformationCost_i), \qquad (2.3)$$

де *RootScore* – відповідність всього тіла;

PartScore – відповідність частини тіла;

DeformationCost – штраф на частину тіла.

Переваги:

− стійка до деформацій та часткового перекриття;

− гнучкіша;

− трохи швидша;

− має розширення на кілька класів.

Недоліки:

− вразлива до складного фону;

− все ще бракує швидкості.

Серед інших графових моделей можна згадати про Markov Random Fields та Conditional Random Fields. З їх переваг це врахування контексту та зв’язків між частинами тіла, глобальна узгодженість та гнучкість до комбінацій. Недоліки це складність навчання, потреба в апроксимації та значні обчислювальні витрати на відео, що робить їх неактуальними.

Найбільш поширеним став спосіб HOG + SVM в умовах статичних зображень. Цього вдалося досягти завдяки тому, що HOG надає важливі ознаки для опису контуру, а SVM забезпечує високу якість класифікації. Треба відмітити, що така реалізація забезпечує високу швидкість навіть на обмежених обчислювальних ресурсах на CPU без потреби в GPU. Навчання можна проводити навіть на невеликих наборах даних. Навіть через широкий

спектр застосування все одно потребується застосування додаткових методів для визначення ключових поз людини.

Розвиток глибокого навчання призвів до появи конволюційних нейронних мереж (CNN) та моделей трансформерів [20]. У загальному порівнянні CNN є швидшими, менші вимоги до пристроїв, не така висока точність, але все ще добра, широка підтримка. Тому далі у цьому розділі будуть розглядатися тільки CNN моделі.

Завдяки появі конволюційних нейронних мереж стала можливою точна та швидка розмітка ключових точок на тілі людини у режимі реального часу. З різновидів стали виділяти моделі для визначення однієї людини та кількох на зображенні. У контексті даної роботи розглядаються моделі для однієї людини.

DeepPose – перша відома глибока нейронна мережа на базі CNN для точного визначення координат ключових точок [21]. Вона навчалася на дуже великому наборі даних з мітками. Серед відомих проблем є велика кількість помилок при визначенні поз, особливо в складних умовах. Мережа є багатоетапною. На першому етапі передбачаються приблизні координати важливих точок. На наступних відбувається їх уточнення. Зазвичай мережа складається з вхідного шару, містить від 4 до 6 конволюційних шарів, шари підсумовування (між конволюційними), від 2 до 3 повнозв’язних шарів та вихідний шар. Вхідний шар потрібен для подання н вхід зображення. Конволюційні шари обробляють зображення для виділення важливих ознак та мають різну кількість фільтрів у порядку збільшення їх кількості. Шари підсумовування слугують для зменшення кількості параметрів для спрощення обчислень. Повнозв’язні шари обробляються отримані ознаки та прогнозують координати ключових точок. На вихідному етапі отримуємо вихідні нейрони у якості карт активацій (для зображень виступають координатами). Функцією втрат при навчанні виступає евклідова функція. Вона зазвичай визначається як квадрат різниці між передбаченим значенням та справжнім значенням (міткою) для кожного зразка:

$$L = \sqrt{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \quad (2.4)$$

де y_i – справжнє значення;

$\hat{y}_i$ – передбачене значення.

Функція втрат необхідна для оцінки ефективності роботи нейронної мережі. За допомогою неї модель визначає як потрібно коригувати параметри (ваги) для покращення результатів. У більшості мереж використовується градієнтний спуск. Його суть полягає у обчисленні градієнта на основі похідної функції втрат параметра (отримаємо напрямок збільшення функції втрат), після чого оновлюємо параметри в протилежному напрямку. Таким чином досягається мінімізація помилки.

Переваги DeepPose:

− прямий підхід до прогнозування без додаткових обчислень;

− масштабованість;

− використання глибоких мереж;

− робота в реальному часі.

Недоліки DeepPose:

− великі вимоги до обчислювальних ресурсів;

− проблеми з точністю в складних умовах;

− обмеження при роботі в реальному часі.

Наступним кроком у нейронних мережах стала OpenPose [22]. Вона має більш складну архітектуру, що використовує Path Affinity Fields. Це дозволило точніше відстежувати зв’язки між частинами тіла та коректного зв’язування точок. У мережі застосовується функція втрат для теплових карт та окрема для конволюційних шарів. Це дозволило набагато підвищити точність. Через зменшену кількість параметрів вдалося досягти й підвищення швидкості архітектури. Також були додаткові оптимізації як покращення використання пам’яті та підтримка прискорення GPU.

PoseNet – це модель для оцінки поз, розроблена Google [23]. Є однією з найбільш доступних і простих у використанні, зокрема підтримці в

TensorFlow. Це дозволило запускати її на різних платформах, зокрема в браузері. PoseNet є конволюційною нейронною мережею, що базується на MobileNet, тому не споживає багато ресурсів і має оптимізацію для роботи на мобільних пристроях. Модель можливо запускати безпосередньо на кінцевих пристроях. Навіть зараз ця модель є досить поширеною у використанні. Через меншу складність моделі знизилася і її точність, але було отримано великий виграш у швидкодії, що дозволило запускати її у реальному часі. Наразі її використовують в інтерактивних іграх, доповненій реальності та розпізнаванні рухів.

Варто згадати про BlazePose [24]. Це потужна модель розроблена Google для MediaPipe. Мережа має двохетапну архітектуру: BlazeDetector, BlazePose Estimator. Перший етап локалізує людину, а другий визначає ключові точки. Застосовується принцип Loghtweight CNN подібний до MobileNet, але ліпше оптимізований. Все це дозволило довести продуктивність у реальному часі до високого рівня та зберегти високу точність. Також модель визначає 33 ключові точки.

MoveNet – модель на базі CNN від Google [25]. Головні особливості – швидкість та точність. Серед усіх розглянутих моделей ця є найшвидшою. Наявні версії для однієї та декількох людей. Є полегшена версія Lightning та складніша Thunder. На Lightning досягається найвища швидкість, але менша точність. Thunder дає вищу точність, але нижчу швидкість. На виході формується 17 ключових точок для кожної людини. Модель є простою у інтеграції, так як є в TensorFlow, та має малу вагу. У якості основи мережі використовується MobileNetV2. Обробка складається з одного етапу. Архітектура мережі складається з 12-14 блоків, які складаються з 2-3 послідовностей Conv2D чи DepthwiseConv2D, ReLU6 (не в усіх) та BatchNorm. Такі блоки називають інвертованими резидуальниими. Conv2D перетворюють зображення у карту ознак за допомогою обчислення згортки між даними на вході і ядром (фільтром). Це потрібно для витягання локальних патернів (контурів, текстур). ReLU6 потрібно використовувати для досягнення

нелінійності та фільтрування негативних значень з обмеженням максимального значення до 6. BatchNorm виконує функцію нормалізації активації шару. Це треба для нульового середнього значення та одиничної дисперсії. DepthwiseConv2D виконує окрему згортку для кожного каналу зображення і при цьому значно зменшує обчислювальні витрати зі збереженням просторової інформації (у порівнянні з Conv2D зменшення відбувається приблизно у 9 разів, залежить від конкретного випадку). Така зв’язка забезпечує ефективне виділення ознак, забезпечує хорошу глибину мережі та мінімальне падіння продуктивності.

Таким чином, серед усіх перерахованих способів оцінки поз на основі побудови ключових точок тіла людини було обрано MoveNet. Так як він базується на CNN архітектурі, то це забезпечує масштабованість та легкість у застосуванні. Ця мережа є найшвидшою серед усіх розглянутих, підтримується TensorFlow у браузері та має високу точність. Для досягнення вищої точності у разі потреби можна перейти з моделі Lightning на Thunder, але доведеться зазнати втрат у швидкодії обробки. Якщо виникне потреба переходу вебзастосунку на відстеження декількох людей, то такий режим тут теж присутній. У зв’язку з тим, що TensorFlow має вбудовані модифікації цієї моделі, то такий код буде легко підтримувати. Також завдяки вбудованими обчислювальними двигунами у цю бібліотеку вдасться досягти максимально можливої швидкості роботи програми у браузері з мінімальними затримками у реальному часі навіть на мобільних пристроях без потреби у введенні додаткових оптимізацій.

2.2 Особливості методів контролю правильності виконання фізичних вправ у реальному часі

Важливість контролювання техніки виконання вправ є невід’ємною частиною збереження здоров’я людини та його покращення. Багато людей за

відсутності досвіду та тренера можуть випадково травмувати себе навіть при виконанні базових вправ у неправильній техніці. Для запобігання таким травмам людина має бачити свої помилки при виконанні та отримувати зворотній зв’язок для їх виправлення. Саме для цього і застосовуються методи для відслідковування техніки виконання.

Контроль за правильністю виконання фізичних вправ у реальному часі є складною задачею. Першочерговим кроком є встановлення обмежень щодо умов у яких буде використовуватися обрана конволюційна нейронна мережа MoveNet.

Обмеження:

− у кадрі має бути лише одна людина;

− людину на зображенні має бути повністю видно;

− місцевість перебування людини повинна мати достатній рівень освітлення.

Виходячи з цього переліку, слід застосовувати Single-pose версію моделі. Для кращої точності варто обрати модифікацію Thunder. При наявних обмеженнях вона буде давати високий рівень точності та забезпечувати належну швидкість роботи вебзастосунку.

Для роботи моделі як вхідні дані будуть почергово передаватися кадри з камери. Після отримання інформації мережа обробить її та дасть на вихід 17 ключових точок, кожна з яких складається з координат x, y та назви частини тіла, додатково буде прописано ймовірність її приналежності, тому можливо здійснювати фільтрацію точок, які слід відображати, а які – не слід, на основі значення граничного рівня точності (рис. 2.1).

Граничний рівень точності слід підбирати індивідуально емпіричним шляхом допоки не буде досягнуто потрібного результату. У даній роботі нижній рівень точності було прийнято за 0,4.

Важливі точки є фіксованими анатомічними точками на тілі людини. Це основа для аналізу положення тіла, руху та структури скелета.

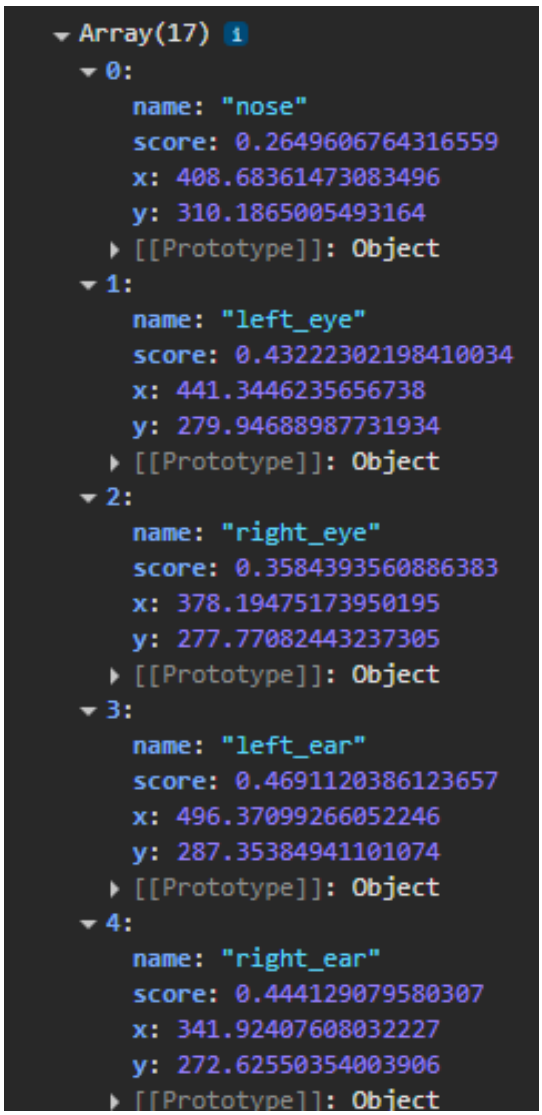


Рисунок 2.1 – Приклад формату виводу ключових точок у консолі

Ці 17 точок складаються з голови (ніс, два ока, два вуха), плечей, ліктів, зап’ясть, стегон, колін та щиколоток. Якщо між цими точками додатково провести лінії, то можливо отримати примітивну модель людини (рис. 2.2).

За допомогою таких точок можливо визначити позу людини чи проаналізувати рух. Для здійснення відстеження дотримання правильної техніки у фізичних вправах обраховують кути між суглобами (ключовими точками) або відстаней чи навіть поєднують обидва способи. Еталонним виконанням вважається дотримання середніх значень кутів та відстаней отриманих від експертів. Завдяки можливості відстеження траєкторії руху можна автоматично надавати додаткові рекомендації людині щодо виконання вправ.

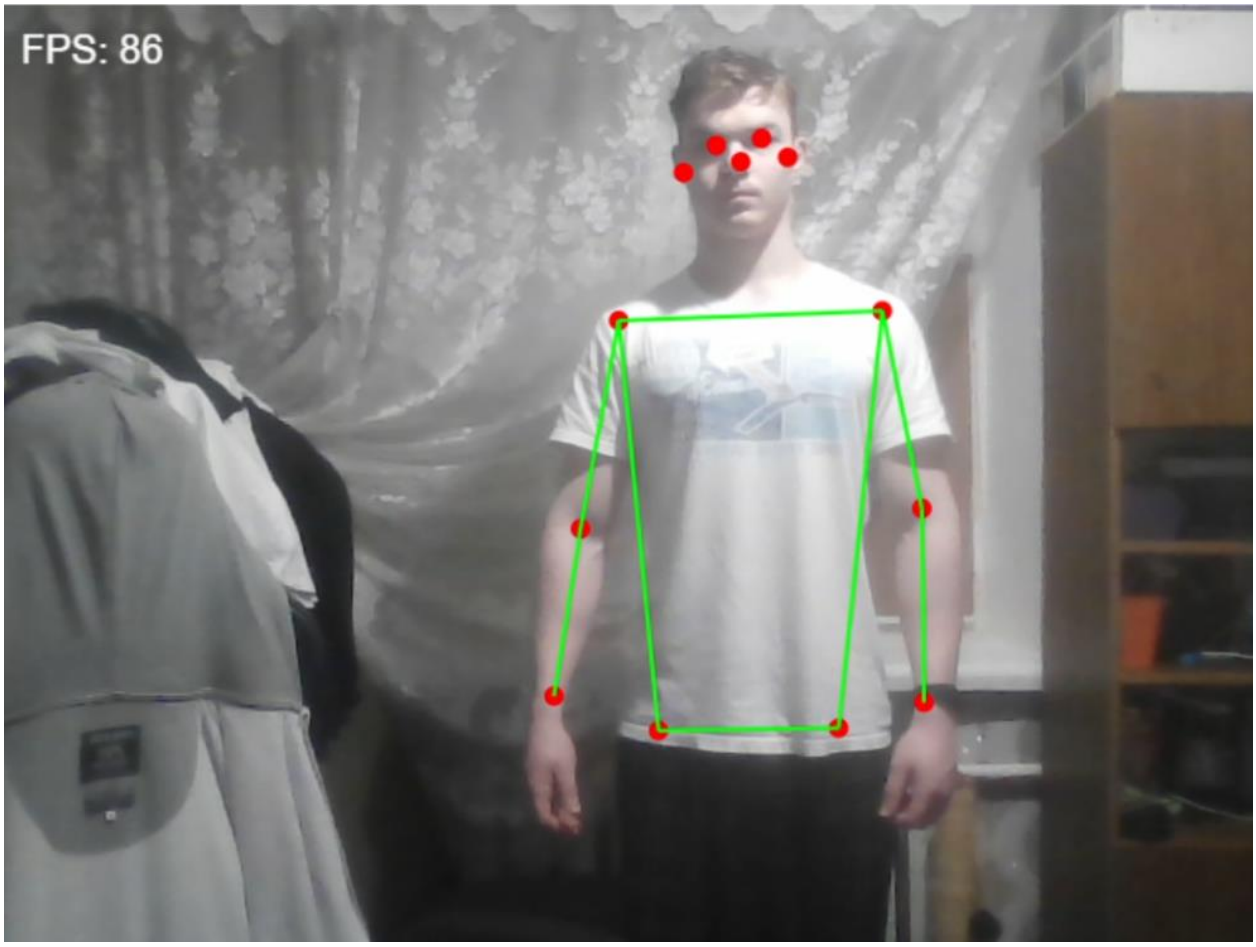


Рисунок 2.2 – Демонстрація об’єднання визначених ключових точок та спроєктованих на відео

2.3 Формулювання методики розпізнавання та контролювання правильності виконання фізичних вправ у реальному часі

Для вебзастосунку було поставлене завдання реалізувати три базові вправи, які можна виконувати з власною вагою – це відтискання, присідання та скручування на прес. У комплексному підході вони дозволяють задіяти всі основні групи м’язів, що дозволяє підтримувати чи навіть покращувати загальний стан організму, так як підтримка свого тіла у хорошому фізичному стані є невід’ємною частиною життя кожної людини.

Так як модель дає на вихід 17 контрольних точок, цього цілком достатньо для контролювання та зарахування виконання вправ.

Варто лише прописати логіку контролю для кожної фізичної вправи окремо на основі положень частин тіла людини. Для цього застосуємо обчислення кутів та відстаней між координатами контрольних точок.

Розрахунок кутів між частинами тіла будемо проводити за наступною функцією (рис. 2.3).

```
const vectorLength = (a) => {
  return Math.sqrt(a.x ** 2 + a.y ** 2);
};

const toVector = (p1, p2) => ({
  x: p2.x - p1.x,
  y: p2.y - p1.y,
});

const calculateAngle = (a, b, c) => {
  const vectorBA = toVector(b, a);
  const vectorBC = toVector(b, c);

  const dotProduct = vectorBA.x * vectorBC.x + vectorBA.y * vectorBC.y;
  const lengthBA = vectorLength(vectorBA);
  const lengthBC = vectorLength(vectorBC);

  const cosineAngle = dotProduct / (lengthBA * lengthBC);
  let angle = Math.acos(cosineAngle);
  angle = (angle * 180) / Math.PI;

  return angle;
};
```

Рисунок 2.3 – Скриншот функції для обчислення кутів між частинами тіла

Функція розраховує кут між трьома точками. Додатково прописано метод для створення вектору з двох точок:

$$\overrightarrow{AB} = \begin{pmatrix} x_B - x_A \\ y_B - y_A \end{pmatrix}. \tag{2.5}$$

Його застосовуємо для створення векторів BA та BC. Далі обчислюємо скалярний добуток цих векторів:

$$\vec{A} \cdot \vec{B} = x_A \cdot x_B + y_A \cdot y_B. \tag{2.6}$$

Також у окремій функції реалізовано знаходження довжини векторів. Застосуємо її до кожного вектору:

$$\left|\vec{A}\right| = \sqrt{{x_A}^2 + {y_A}^2}. \quad (2.7)$$

На основі цих значень можемо знайти косинус кута між векторами за формулою з геометрії:

$$cos\left(\widehat{\vec{A}, \vec{B}}\right) = \frac{\vec{A} \cdot \vec{B}}{|\vec{A}| \cdot |\vec{B}|}. \quad (2.8)$$

Далі за допомогою арккосинуса та геометричних перетворень отримаємо кінцевий результат у градусах.

Перед тим як описувати вимоги для кожної вправи, треба розглянути загальний підхід до взяття координат. Це важливо, тому що моделі CNN не є ідеальними та не обраховують справжні координати, а лише наближені. Так як ми задаємо мінімальний рівень точності, то якісь ключові точки просто будуть відкинуті у процесі обробки. Через це виникає необхідність у перевірки наявних точок. Для обробки потрібно мати мінімум 6 точок з одного боку, у найкращому сценарії будемо мати 12.

6 точок через те, що це зап’ястя, лікоть, плече, стегно, коліно та щиколотка з одного боку людини. Якщо системі вдається одночасно точно розпізнати два боки, то отримаємо 12 точок. Підхід обробки точок з лівої частини тіла, з правої чи з усього разом, забезпечує вищу результативність і зменшує залежність від точності моделі, у порівнянні з обробкою лише усіх точок разом.

Опис вимог для відтискань.

Перевірки:

− розташування долонь на підлозі: розраховуємо середнє арифметичне для зап’ясть та окремо для стегон. Стегна мають бути вищими за зап’ястя;

− розташування долонь на ширині плечей: обчислюємо кут між стегном, плечем та ліктем. Цей кут повинен бути менше або дорівнювати 90°;

− згинання ліктів під кутом 45°: знаходимо максимальні значення за ординатою для ліктів та зап’ясть. Лікті мають знаходитися вище;

− рівність тулуба: розраховуємо вектор між стегном і плечем. Використовуємо функцію Math.atan2 для обчислення кута, тангенс якого дорівнює відношенню абсциси до ординати. Кут переводимо у градуси та порівнюємо зі значенням 100°. Якщо тулуб рівний, то отримане значення має бути нижчим;

− розташування ніг на ширині плечей: знаходимо максимальні значення за ординатою для щиколоток та зап’ясть. Щиколотки повинні розташовуватися вище;

− початкова позиція: обчислюємо кут між зап’ястям, ліктем та плечем. Кут має бути більшим або дорівнювати 160°;

− кінцева позиція: обчислюємо кут між зап’ястям, ліктем та плечем. Кут має бути меншим або дорівнювати 100°.

Під час перевірок формуються підказки для користувача, які виводяться у окремому блоці (рис. 2.4).

Варто зауважити, що для зарахування вправи головними чинниками є початкова та кінцева позиція, інші перевірки не впливають на підрахунок кількості виконаних разів. Це було зроблено для ширшого застосування вправи. Дозволено виконувати вправу з колін, у цьому випадку не слід звертати увагу на підказку про розташування тазу відносно плечей. Якщо виконуєте вправу з вузькою постановкою рук, можна знехтувати підказкою про ширину постановки ніг або поставити їх вужче. Враховано лише критичні відхилення від техніки, такі як згинання ліктів та розташування долонь. На додачу, це дозволяє стояти людині у різних положеннях відносно камери, хоч лівою, хоч правою стороною. Обличчям та спиною під час виконання вправи до камери ставати не варто, бо застосунку буде бракувати інформації про деякі кути між частинами вашого тіла.

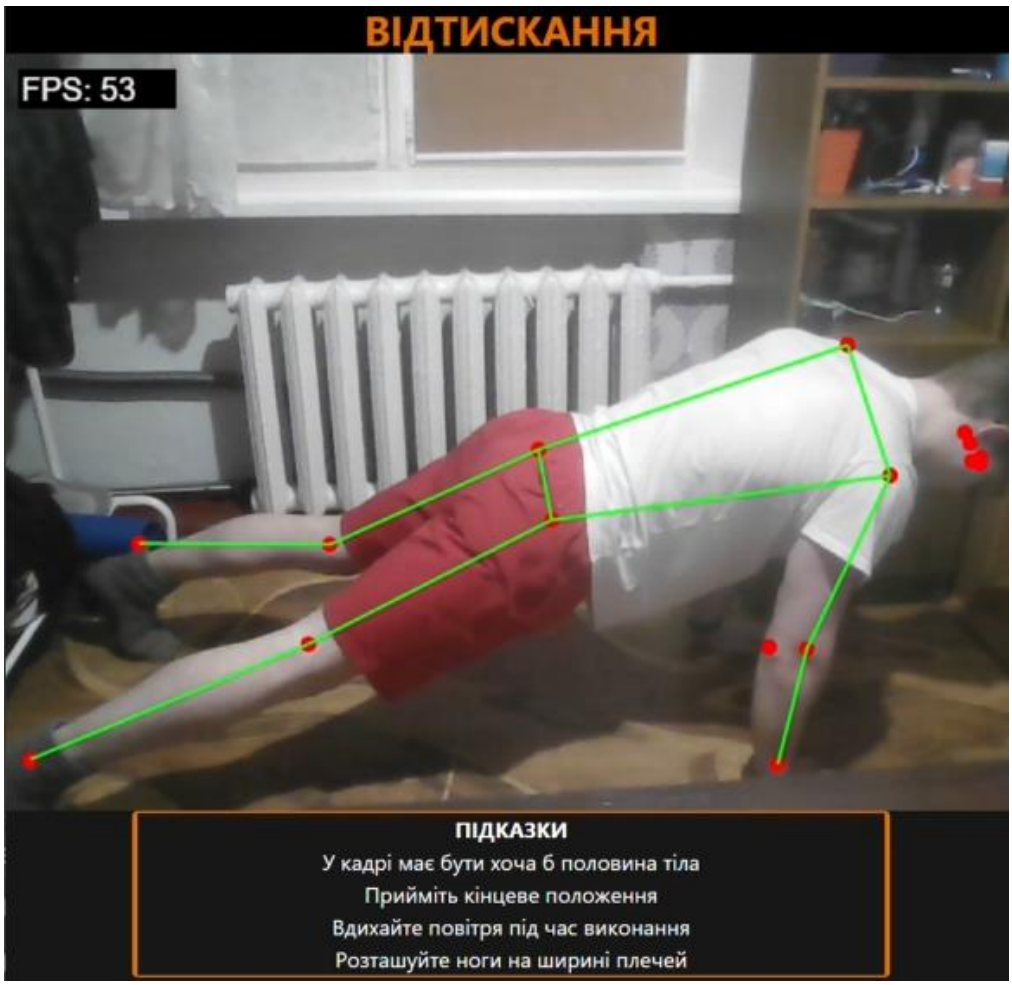


Рисунок 2.4 – Скриншот роботи вебзастосунку для відтискань на момент розробки

Опис вимог для присідань.

Перевірки:

− нахил спини: обчислюємо компоненти вектору між плечем та стегном, потім застосовуємо вбудовану функцію Math.atan2 для знаходження кута, тангенс якого дорівнює відношенню абсциси до ординати. Знайдений кут переводимо з радіан у градуси та порівнюємо відхилення (максимальне відхилення було задано як 50°);

− розташування колін відносно носків та п’ят: обчислюємо різницю за модулем між положенням коліна та щиколотки, перевіряємо чи більша вона за половину частини ноги між стегном та коліном;

− початкова позиція: обчислюємо кут між стегном, коліном та щиколоткою. Кут має бути не меншим за 160°;

– кінцева позиція: обчислюємо кут між стегном, коліном та щиколоткою. Кут повинен бути меншим або дорівнювати 90°.

Підказки формуються спираючись на описані перевірки (рис. 2.5).

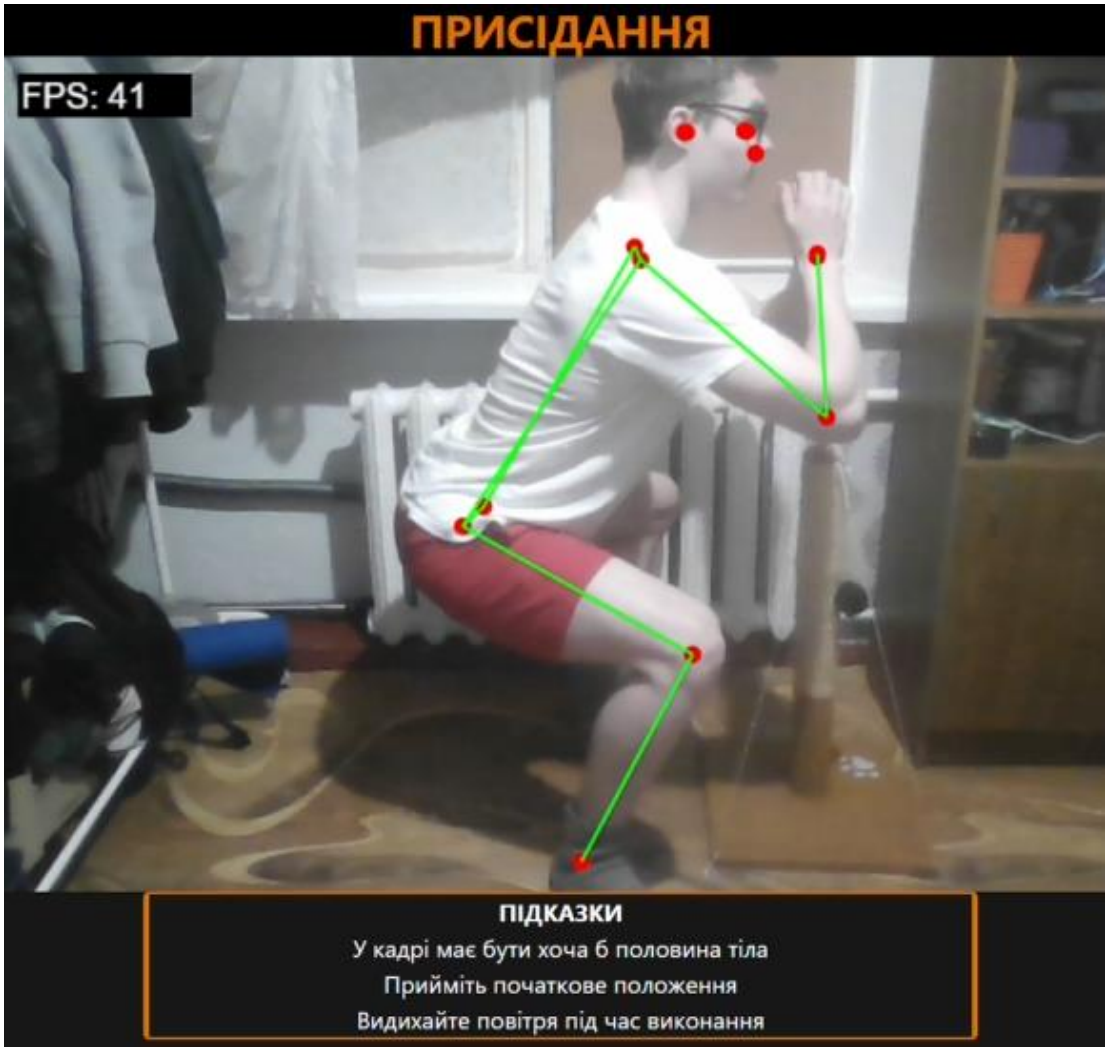


Рисунок 2.5 – Скриншот роботи вебзастосунку для присідань на момент розробки

Опис вимог для скручувань на прес.

Для усіх перевірок цієї вправи застосовується головна перевірка. Вона полягає у тому, що обчислюються координати положення плечей за ординатою як середнє арифметичне і так само для колін. Плечі повинні бути нижчими за коліна. Тільки за цієї умови будуть відбуватися усі подальші перевірки, якщо умову не буде виконано, то виконання вправи не буде зараховуватися.

Для початкової і кінцевої позиції варто знайти кут відносно ординати. Щоб дізнатися його значення, потрібно обчислити вектор між плечем та стегном. Наступним кроком використовуємо Math.atan2 для отримання кута, тангенс якого дорівнює відношенню абсциси до ординати. Переводимо кут у градуси.

Перевірки:

− кут в колінах: обчислюємо кут між стегном, коліном та щиколоткою. Кут має бути меншим чи дорівнювати 90°;

− початкова позиція: беремо кут відносно ординати. Він повинен бути менше чи дорівнювати 90°;

− кінцева позиція: беремо кут відносно ординати. Кут має бути більшим або дорівнювати 110°.

Для скручувань підказки формуються аналогічним чином до минулих вправ (рис. 2.6).

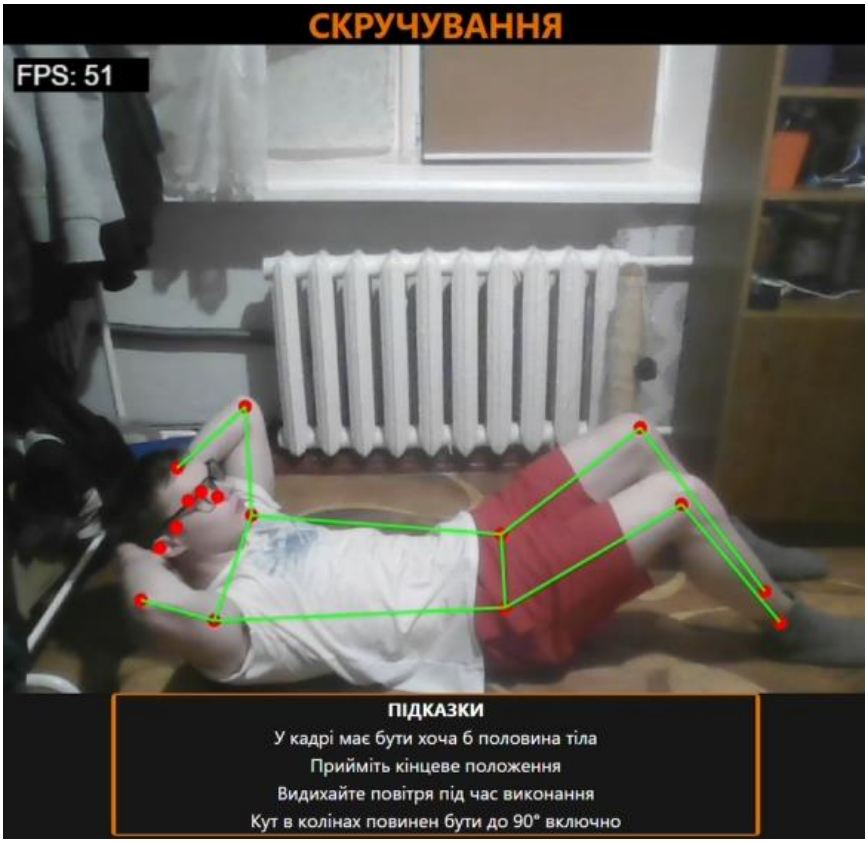


Рисунок 2.6 – Скриншот роботи вебзастосунку для скручувань на момент розробки

Варто зауважити, що для кожної вправи існує специфічний перечень положень, у яких їх варто виконувати. Для усіх вправ це або положення тіла людини лівою стороною до камери або правою. Якщо стати обличчям до камери чи спиною, то програмі бракуватиме інформації про деякі кути потрібні для перевірки вимог щодо виконання вправ, тому використання цих поз не є можливим для коректної роботи застосунку.

Точність детекції значною мірою залежить від положення тіла людини, відстані, апаратури, освітлення та навіть одягу, тому при виконанні вправ потрібно звертати увагу на підказки та дотримуватися їх для отримання оптимального рівня роботи вебзастосунку. Мають сенс експерименти з застосуванням застосунку у різному одязі, у приміщеннях з різним рівнем освітлення чи навіть на дворі. Якщо бажаєте досягти оптимального рівня детекції, бажано записувати вправу без людей на фоні та з мінімумом рухомих об’єктів, які можуть завадити алгоритму з виявлення головних ключових точок. Також не варто вдягати одяг темних кольорів в погано освітлених приміщеннях.

2.4 Моделювання структури вебзастосунку для розпізнавання та контролювання правильності виконання фізичних вправ у реальному часі

Для вдалої реалізації вебзастосунку необхідно правильно сформувати його структуру, у яку входять опис процесів та взаємодія класів. Створення діаграм такого типу забезпечує інструментальний засіб під назвою Rational Rose.

У формуванні програмного забезпечення буде застосовуватися UML, щоб мати можливість візуального моделювання поведінки та взаємодії між компонентами. З основних типів діаграм цієї стандартизованої мови, будемо використовувати діаграму прецедентів та діаграму класів.

Діаграма прецедентів – один з інструментів UML, що використовується для візуалізації системи з точки зору користувачів. Прецедент – окрема функція, яку система виконує у відповідь на взаємодію з актором. Актор – зовнішній учасник, який взаємодіє з системою. Між прецедентами та акторами створюють зв'язки для відображення взаємодії.

Опис комунікативної комп'ютерної системи для здійснення розпізнавання та контролювання правильності виконання фізичних вправ у реальному часі:

− щоб скористатися можливостями комунікативної комп'ютерної системи, користувачам не потрібно проходити етап ідентифікації;

− користувач має можливість переходити до тренування, відкривати інструкцію до вправи, скидати прогрес з вправи, починати тренування, змінювати налаштування та завершувати процес тренування.

Опис акторів в комунікативній комп'ютерній системі для здійснення розпізнавання та контролювання правильності виконання фізичних вправ у реальному часі:

− користувач – людина, якій надається право використовувати певні функціональні можливості, що надаються системою;

− вебзастосунок – застосунок, що дозволяє здійснювати управління комп'ютерною системою;

− система відстеження рухів людини – система, що дозволяє подавати на вхід дані про поточний прогрес вправи та зображення з камери для їх автоматичного оброблення та формування вихідних даних.

Опис варіантів використання (прецедентів) в комунікативній комп'ютерній системі для здійснення розпізнавання та контролювання правильності виконання фізичних вправ у реальному часі:

− перейти до тренування – цей варіант використання ініціюється користувачем. Забезпечує можливість відображення компонента тренування у системі;

– відкрити інструкцію – цей варіант використання ініціюється користувачем. Забезпечує можливість відображення модального вікна з інструкцією виконання щодо конкретної вправи у системі;

– скинути прогрес – цей варіант використання ініціюється користувачем. Забезпечує можливість скидання прогресу з конкретної вправи у системі;

– почати тренування – цей варіант використання ініціюється користувачем. Забезпечує можливість запуску процесу тренування у системі;

– відкрити налаштування – цей варіант використання ініціюється користувачем. Забезпечує можливість відображення модального вікна з налаштуваннями у системі;

– завершити тренування – цей варіант використання ініціюється користувачем. Забезпечує можливість завершення процесу тренування у системі;

– запустити систему – цей варіант використання ініціюється вебзастосунком. Забезпечує можливість запуску системи відстеження рухів людини;

– оновити налаштування – цей варіант використання ініціюється вебзастосунком. Забезпечує можливість оновлення налаштувань системи відстеження рухів людини;

– передати вхідні дані – цей варіант використання ініціюється вебзастосунком. Забезпечує можливість передачі вхідних даних із застосунку до системи відстеження рухів людини;

– отримати вихідні дані – цей варіант використання ініціюється системою відстеження рухів людини. Забезпечує можливість передачі вихідних даних з системи відстеження рухів людини до вебзастосунку.

Список акторів та варіантів використання (прецедентів), описаної комунікативної комп’ютерної системи, зображено на рисунку 2.7. Головна діаграма варіантів використання описаної комунікативної комп’ютерної системи зображено на рисунку 2.8.

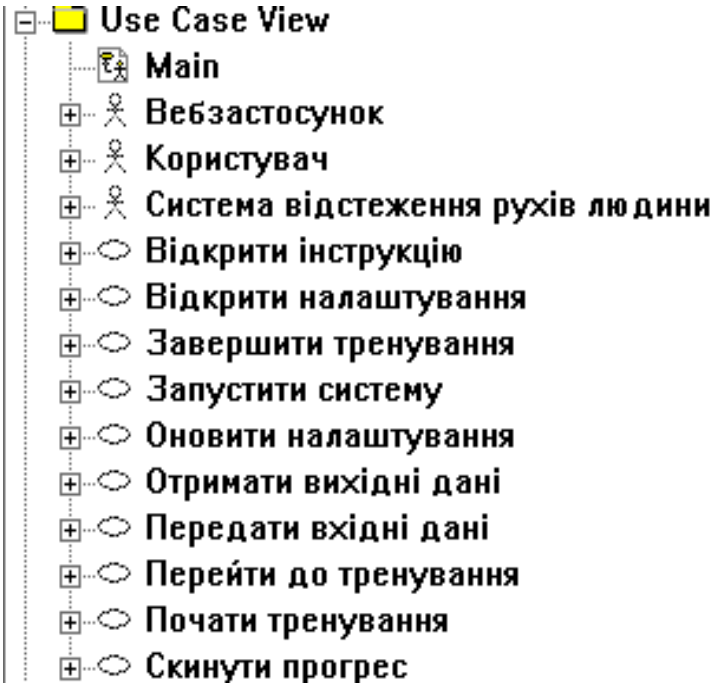


Рисунок 2.7 – Список акторів та прецедентів системи здійснення розпізнавання та контролювання правильності виконання фізичних вправ у реальному часі

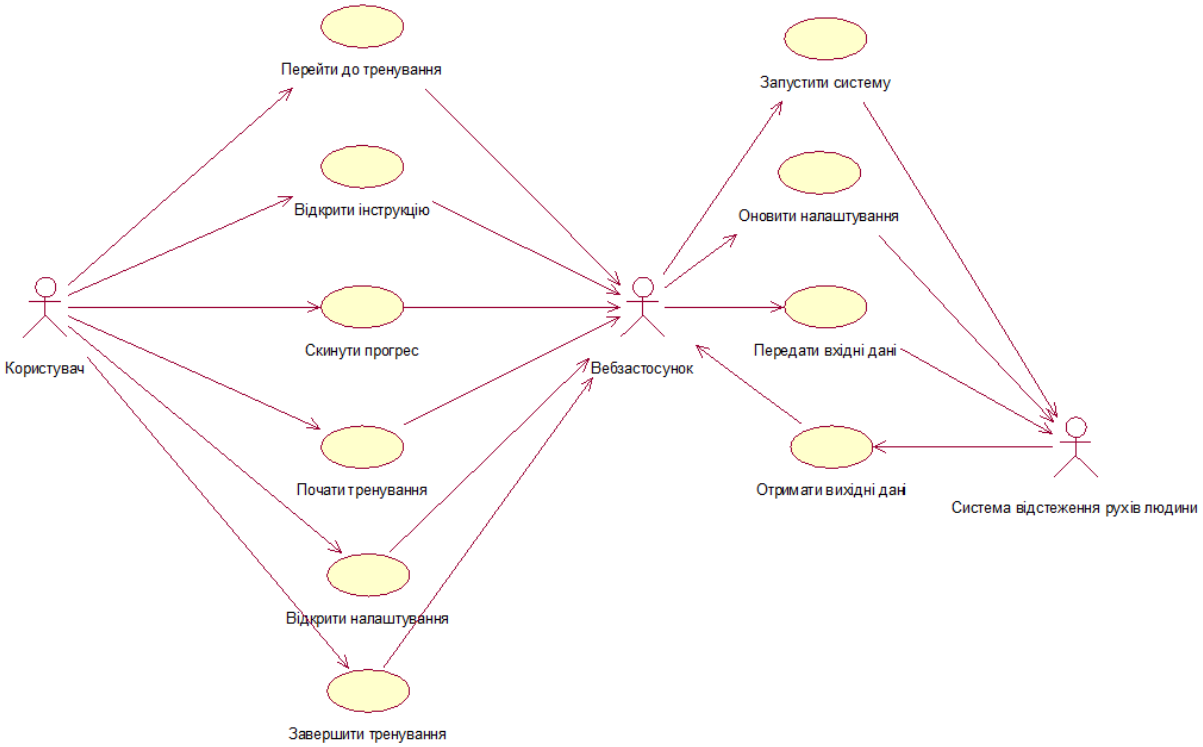


Рисунок 2.8 – Головна діаграма варіантів використання системи для здійснення розпізнавання та контролювання правильності виконання фізичних вправ у реальному часі

Діаграма класів – один з інструментів UML, що використовується для моделювання статичного виду системи. На такій діаграмі відображають, як відбуваються взаємодії, але не їх здійснення. Вона складається з класів, інтерфейсів, об’єктів, кооперацій та відношень. Може також мати примітки, обмеження, пакети, підсистеми чи екземпляри. Клас – опис сукупності об’єктів, що містять загальні атрибути, операції, відношення, семантику.

Діаграма класів для описаної комунікативної комп’ютерної системи зображено на рисунку 2.9.

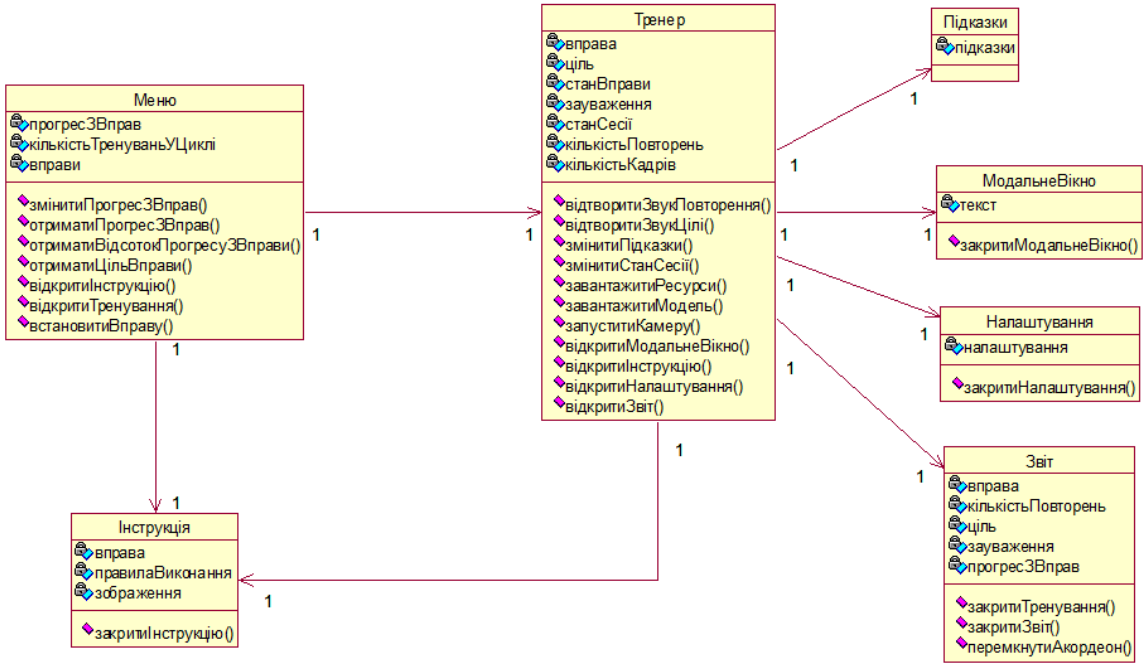


Рисунок 2.9 – Діаграма класів системи для здійснення розпізнавання та контролювання правильності виконання фізичних вправ у реальному часі

Окрему увагу варто приділити процесу формування прогресу з виконання вправ. Заданий підхід являє собою мікроцикл технічної адаптації, оскільки розрахований на виконання лише одного підходу за тренування. Застосовується лінійна періодизація – коли навантаження поступово зростає у циклі. На початку та у кінці кожного циклу присутнє пікове тренування – коли людиною виконується максимально можлива кількість повторень вправи за тренування.

Такий підхід дозволяє:

− поступово збільшувати інтенсивність, що знижує ризик травм;

− підходить для новачків і середнього рівня;

− дає змогу планомірного відстеження прогресу.

Серед недоліків:

− один підхід може не дати необхідного рівня навантаження для стимулу прогресу;

− для просунутих спортсменів краще застосовувати нерівномірно періодизовані цикли.

Цикл складається з восьми тренувань (табл. 2.1).

Таблиця 2.1 – Мікроцикл лінійної періодизації з піковими тестами

Номер тренування	**Кількість повторень (від % від максимуму)**	**Мета**
1	100%	Визначення максимуму
2	60%	Відновлення
3	75%	Пульс інтенсивності
4	65%	Вибухове виконання
5	85%;	Високе навантаження
6	70%	Відновлення
7	90%	Підготовка
8	100%	Перевірка прогресу

Після закінчення першого циклу результат останнього тренування переноситься як результат першого тренування нового циклу. Такий підхід допомагає уникнути перенавантаження людини. Якщо людина у процесі виконання мікроциклу захоче оновити свій максимум, вона може або скинути прогрес з вправи на головній сторінці у меню, або просто зробити на тренуванні нову пікову кількість повторень. У такому разі система автоматично оновить максимум та оновить навантаження для подальших тренувань.

3 ЕТАПИ РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ ДЛЯ РОЗПІЗНАВАННЯ ТА КОНТРОЛЮВАННЯ ПРАВИЛЬНОСТІ ВИКОНАННЯ ФІЗИЧНИХ ВПРАВ У РЕАЛЬНОМУ ЧАСІ

3.1 Вибір інструментальних засобів реалізації вебзастосунку для розпізнавання та контролювання правильності виконання фізичних вправ у реальному часі

Для розробки вебзастосунку було обрану мову програмування JavaScript [26] у редакторі коду Visual Studio Code. Вона підтримується в усіх сучасних браузерах, на мобільних і десктопах. Є швидкою, зручною, зрозумілою та популярною. Містить широку екосистему бібліотек. Дозволяє додавати інтерактивність до клієнтської частини. Керування пакетами здійснюється за допомогою менеджера Node Package Manager. Він є стандартним інструментом, працює на Node.js і входить до його складу при встановленні. Сам Node.js використовується для запуску вебзастосунків, написаних на JavaScript.

Щоб прискорити процес оформлення сторінки без написання великої кількості CSS коду з нуля, було застосовано Tailwind CSS [27]. Він собою являє фреймворк (набір готових інструментів) CSS, що дозволяє створювати атомарні класи-утиліти. Класи піддаються модифікаціям та надають повну свободу дій.

Переваги фреймворку:

− швидка розробка інтерфейсу;

− мінімальна вага кінцевого файлу;

− повна структура компонента в одному місці;

− зручна підтримка адаптивності;

− стандартизований дизайн;

− гнучкість без обмежень;

− сумісність з іншими фреймворками.

У якості збирача проєкту буде застосовуватися такий інструмент як Vite [28]. Він дозволяє розробнику витрачати мінімум часу на конфігурацію збірки та максимум часу на саму розробку. Його процес оптимізації та використання сучасних можливостей JavaScript дозволяють бачити зміни у вебзастосунку без перезавантаження сторінки у браузері. Застосування цього інструменту надає змогу здійснювати компіляцію фреймворків і бібліотек автоматично при збереженні змін. Vite без проблем підтримує нові технології та легко інтегрується з ними.

З метою полегшення розробки інтерфейсу, було прийнято рішення застосування компонентного підходу. Такий підхід забезпечує оновлення лише зміненої частини сторінки, це робить застосунок швидшим та ефективнішим. Досягається це шляхом застосування віртуального DOM – копії реального DOM. Для цього було застосовано бібліотеку React [29, 30]. Ця бібліотека дозволяє писати HTML-подібний код прямо в JavaScript, що робить код більш читабельним. Підтримка від Meta гарантує стабільність і регулярність оновлень. Наявність великої кількості бібліотек для роботи з React забезпечує додаткову гнучкість у розробці, це дозволяє додавати необхідний функціонал до застосунку без зайвих зусиль.

У зв’язку з потребою постійного передавання станів компонентів один між одним, виникає потреба у застосуванні та використанні єдиного централізованого сховища, в якому будуть зберігатися дані такого типу. Це зменшить залежності між компонентами у застосунку. Тому будемо застосовувати бібліотеку Redux для керування глобальним станом у React-застосунках [31, 32].

Так як необхідно застосовувати нейронну мережу, то потрібна бібліотека з її підтримкою. Такою є TensorFlow.js [33–35]. Вона дозволяє запускати машинне навчання безпосередньо у браузері з використанням JavaScript. Це дозволяє уникнути необхідності використання сервера. Наявність прискорювачів обчислень у бібліотеці робить можливим підтримку швидкості на високому рівні навіть у реальному часі.

3.2 Етапи програмної реалізації розпізнавання та контролювання правильності виконання фізичних вправ у реальному часі

Першим етапом потрібно встановити необхідні бібліотеки і фреймворки у проєкт. Це можна зробити за допомогою Node Package Manager у консолі редактору Visual Studio Code (рис. 3.1). Потрібні пакети та встановлені команди наведено на рисунку 3.2.

```
npm install [<package-spec> ...]
```

Рисунок 3.1 – Команда для встановлення пакета

```
"scripts": {
  "dev": "vite",
  "build": "vite build",
  "lint": "eslint .",
  "preview": "vite preview"
},
"dependencies": {
  "@reduxjs/toolkit": "^2.7.0",
  "@tailwindcss/vite": "^4.1.4",
  "@tensorflow-models/pose-detection": "^2.1.3",
  "@tensorflow/tfjs": "^4.22.0",
  "react": "^19.0.0",
  "react-dom": "^19.0.0",
  "react-redux": "^9.2.0",
  "tailwindcss": "^4.1.4"
},
"devDependencies": {
  "@eslint/js": "^9.22.0",
  "@types/react": "^19.0.10",
  "@types/react-dom": "^19.0.4",
  "@vitejs/plugin-react": "^4.3.4",
  "eslint": "^9.22.0",
  "eslint-plugin-react-hooks": "^5.2.0",
  "eslint-plugin-react-refresh": "^0.4.19",
  "globals": "^16.0.0",
  "vite": "^6.3.0"
}
```

Рисунок 3.2 – Пакети та команди вебзастосунку у файлі package.json

Після встановлення усіх необхідних залежностей, варто видалити непотрібні файли, які було автоматично створено за допомогою Vite. Структура файлів повинна мати вигляд як на рисунку 3.3.

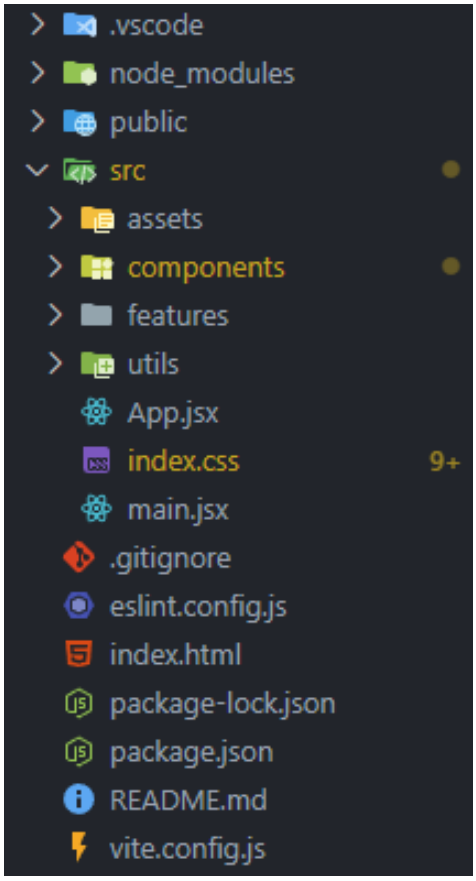


Рисунок 3.3 – Структура файлів застосунку

Деякі сформовані файли потрібно змінити. Одним з таких є файл конфігурації Vite (рис. 3.4). Це важливо для правильної роботи React і Tailwind CSS.

```
1  import {defineConfig} from "vite";
2  import react from "@vitejs/plugin-react";
3  import tailwindcss from "@tailwindcss/vite";
4
5  // https://vite.dev/config/
6  export default defineConfig({
7    plugins: [react(), tailwindcss()],
8  });
```

Рисунок 3.4 – Код файлу з конфігурації Vite

Розглянемо файл HTML (рис. 3.5). У ньому має бути під’єднано модуль main.jsx. Також змінено текст заголовка сторінки.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Physical exercises detector</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

Рисунок 3.5 – Код файлу index.html

Перейдемо до розгляду файлу main.jsx. У ньому відбувається під’єднання файлу з CSS, React до головної сторінки HTML та глобального сховища від Redux до застосунку (рис. 3.6).

```
import {createRoot} from "react-dom/client";
import "./index.css";
import App from "./App.jsx";
import {Provider} from "react-redux";
import {store} from "./utils/store.js";

createRoot(document.getElementById("root")).render(
  <Provider store={store}>
    <App />
  </Provider>
);
```

Рисунок 3.6 – Код файлу main.jsx

Розглянемо вміст папок assets, components, features та utils. Assets містить у собі папку з зображеннями та папку зі звуками (рис. 3.7). Components складається з компонентів застосунку: Hints, Instruction, Menu, Modal, Report,

Settings, Trainer (рис. 3.8). Features складається з папок з частинами коду для керування станами компонентів через глобальне сховище Redux (рис. 3.9). Utils складається з додаткових частин коду для ініціалізації глобального сховища та роботи детектора (рис. 3.10).

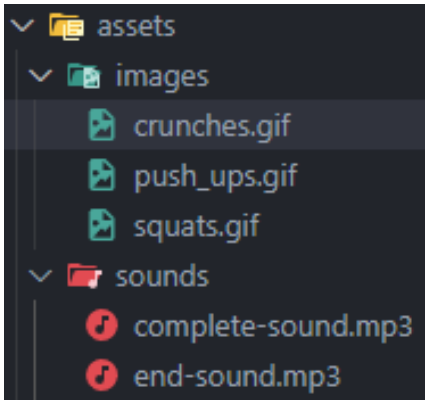


Рисунок 3.7 – Структура папки assets

Рисунок 3.8 – Структура папки components

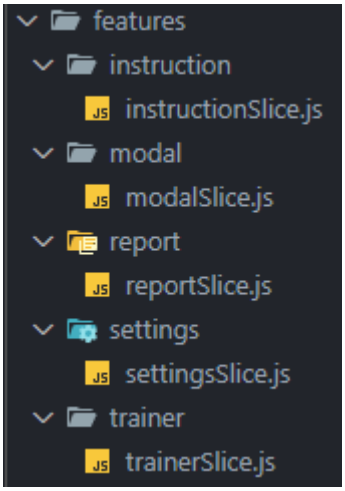


Рисунок 3.9 – Структура папки features

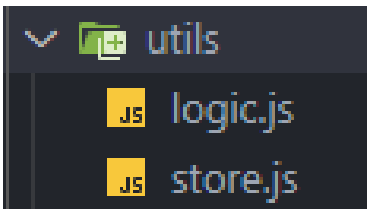


Рисунок 3.10 – Структура папки utils

Перед роботою з компонентами, напишемо код для роботи глобального сховища (рис. 3.11). Воно буде містити інформацію про стани таких компонентів: Modal, Settings, Trainer, Instruction, Report. Кожний з компонентів буде мати свій стан.

```
import {configureStore} from "@reduxjs/toolkit";
import modalReducer from "../features/modal/modalSlice";
import settingsReducer from "../features/settings/settingsSlice";
import trainerReducer from "../features/trainer/trainerSlice";
import instructionReducer from "../features/instruction/instructionSlice";
import reportReducer from "../features/report/reportSlice";

export const store = configureStore({
  reducer: {
    modal: modalReducer,
    settings: settingsReducer,
    trainer: trainerReducer,
    instruction: instructionReducer,
    report: reportReducer,
  },
});
```

Рисунок 3.11 – Код файлу store.js

Код для Modal, Settings, Instruction та Report має різницю лише у назвах, тому приведено лише код до Modal на рисунку 3.12.

```
import {createSlice} from "@reduxjs/toolkit";

const initialState = {
  isModalOpen: false,
};

const modalSlice = createSlice({
  name: "modal",
  initialState,
  reducers: {
    openModal: (state) => {
      state.isModalOpen = true;
    },
    closeModal: (state) => {
      state.isModalOpen = false;
    },
  },
});

export const {openModal, closeModal} = modalSlice.actions;

export default modalSlice.reducer;
```

Рисунок 3.12 – Код файлу modalSlice,js

Код до Trainer маж більшу кількість змінних (рис. 3.13).

```
import {createSlice} from "@reduxjs/toolkit";

const initialState = {
  isTrainerOpen: false,
  exercise: null,
  goal: null,
};

const trainerSlice = createSlice({
  name: "trainer",
  initialState,
  reducers: {
    openTrainer: (state) => {
      state.isTrainerOpen = true;
    },
    closeTrainer: (state) => {
      state.isTrainerOpen = false;
    },
    setExercise: (state, action) => {
      state.exercise = action.payload.name;
      state.goal = action.payload.goal;
    },
  },
});

export const {openTrainer, closeTrainer, setExercise} = trainerSlice.actions;

export default trainerSlice.reducer;
```

Рисунок 3.13 – Код файлу trainerSlice,js

Розглянемо концептуальні моменти роботи застосунку. Відкриття і закриття компонентів відбувається на основі зміни їх станів. Якщо було викликано функцію відкриття з глобального сховища, то стан зміниться. Цю інформацію можна отримати через функцію useSelector. А відображення компонентів відбувається завдяки тернарній операції. Приклад застосування можна побачити на рисунку 3.14.

```
import Trainer from "./components/Trainer";
import Menu from "./components/Menu";
import {useSelector} from "react-redux";

const App = () => {
  const {isTrainerOpen} = useSelector((store) => store.trainer);

  return <>{isTrainerOpen ? <Trainer /> : <Menu />}</>;
};

export default App;
```

Рисунок 3.14 – Код файлу App.jsx

Для виклику функції з Redux сховища треба викликати її через useDispatch. Реалізацію такого коду можна побачити на кнопці в головному меню (рис. 3.15).

```
<button
  type="button"
  onClick={() => {
    dispatch(
      setExercise({
        name: value,
        goal: getExerciseGoal(value),
      })
    );
    dispatch(openTrainer());
  }}
>
  тренування
</button>
```

Рисунок 3.15 – Код реалізації кнопки з запуску тренування у файлі Menu.jsx

Формування прогресу з вправ відбувається з локального сховища вебзастосунку. Якщо у ньому записи відсутні, то формуємо нову шкалу (рис. 3.16).

```
const getExercisesProgress = () => {
  const stored = localStorage.getItem("exercisesProgress");
  const parsed = stored ? JSON.parse(stored) : null;
  const progress = {};
  Object.values(exercises).map((value) => {
    progress[value] = parsed
      ? parsed[value] ?? {max: "?", stage: 0}
      : {max: "?", stage: 0};
  });
  return progress;
};
```

Рисунок 3.16 – Код функції формування прогресу у файлі Menu.jsx

Розрахунок цілей для кожного тренування відбувається динамічним чином під час запуску тренування. З прогресу вправи беремо поточний максимум та номер ітерації циклу. Для першої та останньої ітерації залишаємо ціль як надпис про необхідність виконати вправу на максимальну кількість повторень. Інші ітерації обчислюються як поточний максимум помножений на коефіцієнт відповідний номеру ітерації округлений до найближчого більшого числа (рис. 3.17).

```
const getExerciseGoal = (exercise) => {
  const currentStage = exercisesProgress[exercise].stage;
  if (currentStage === 0 || currentStage === 7) {
    return "максимум";
  } else {
    const currentMax = exercisesProgress[exercise].max;
    const coefficients = {
      1: 0.6,
      2: 0.75,
      3: 0.65,
      4: 0.85,
      5: 0.7,
      6: 0.9,
    };
    return Math.max(Math.ceil(currentMax * coefficients[currentStage]), 1);
  }
};
```

Рисунок 3.17 – Код функції з розрахунку цілі для вправи з файлу Menu.jsx

З метою раціонального використання ресурсів, деякі функції виконуються лише при появі компонента. Це реалізується за допомогою useEffect (рис. 3.18).

```
useEffect(() => {
  setExercisesProgress(getExercisesProgress());
}, []);
```

Рисунок 3.18 – Код з використанням useEffect для отримання прогресу з вправ при відображенні з компоненту Menu

Скидання прогресу з конкретної вправи відбувається шляхом перезапису старого значення та оновлення інформації у локальному сховищі (рис. 3.19).

```
<button
  type="button"
  onClick={() => {
    setExercisesProgress((prevValue) => {
      const newValue = {
        ...prevValue,
        [value]: {"max": "?", stage: 0},
      };
      localStorage.setItem(
        "exercisesProgress",
        JSON.stringify(newValue)
      );
      return newValue;
    });
  }}
>
  скинути
</button>
```

Рисунок 3.19 – Код реалізації кнопки зі скидання прогресу для вправи з файлу Menu.jsx

Розглянемо компонент Trainer. У ньому було реалізовано функцію для доступу до камери (рис. 3.20). Якщо камера відсутня у системі, виникла помилка чи користувач відмовляється надавати доступ, то виводиться компонент Modal з відповідним попередженням.

```
const startWebcam = async () => {
  try {
    const stream = await navigator.mediaDevices.getUserMedia({video: true});
    videoRef.current.srcObject = stream;
    console.log("✅ Камеру успішно підключено.");
  } catch (err) {
    console.error("❌ Помилка доступу до камери:", err);
    setHints(
      "❌ Помилка доступу до камери\nПереконайтеся, що пристрій підключено та не вимкнено!\n"
    );
    dispatch(openModal());
  }
};
```

Рисунок 3.20 – Код функції з налаштування камери у файлі Trainer.jsx

Завантаження аудіо файлів та налаштувань з локального сховища зроблено окремо (рис. 3.21).

```
const loadResources = () => {
  soundExerciseEndRef.current = new Audio(soundExerciseEnd);
  soundExerciseGoalRef.current = new Audio(soundExerciseGoal);

  const defaultSettings = {
    showFPS: true,
    showSkeleton: true,
    showKeypoints: true,
    playSounds: true,
  };

  const storedSettings = localStorage.getItem("settings");
  settingsRef.current = storedSettings
    ? JSON.parse(storedSettings)
    : defaultSettings;
};
```

Рисунок 3.21 – Код функції завантаження ресурсів у компоненті Trainer

Особливої уваги заслуговує процес ініціалізації моделі MoveNet. Для високої швидкості є необхідність запуску машинного навчання на ефективному обчислювальному рушії. Серед таких використовується WebGPU, WebGL, WASM (WebAssembly) та CPU. WebGPU являє собою покращену версію WebGL, але може не підтримуватися деякими пристроями. У такому разі застосовується WebGL. Якщо з якоїсь причини його теж не вдається запустити, то переходимо на бінарний формат WASM. У найгіршому випадку доведеться робити усі обчислення на CPU, що є дуже повільно, оскільки не застосовуються обчислювальні оптимізації, але буде працювати на усіх пристроях.

Після встановлення обчислювального рушія варто обрати саму модель. У вебзастосунку буде використовуватися MoveNet SinglePose Thunder. Якщо виникнути помилки при завантаженні, то про це буде виведено інформацію до консолі.

Весь цей процес описано на рисунку 3.22.

```
const loadModel = async () => {
  const tryBackends = ["webgpu", "webgl", "wasm", "cpu"];

  for (const backend of tryBackends) {
    try {
      await tf.setBackend(backend);
      await tf.ready();

      console.log(
        `✅ TensorFlow.js: обчислювальний рушій встановлено на: ${backend}`
      );

      const detectorConfig = {
        modelType: poseDetection.movenet.modelType.SINGLEPOSE_THUNDER,
      };
      const detectorInstance = await poseDetection.createDetector(
        poseDetection.SupportedModels.MoveNet,
        detectorConfig
      );
      console.log("✅ TensorFlow.js: модель успішно завантажена.");
      setDetector(detectorInstance);
      return;
    } catch (err) {
      console.error(
        `❌ TensorFlow.js: Помилка з бекендом ${backend}.`,
        err.message
      );
    }
  }

  console.error(
    "❌ TensorFlow.js: не вдалося ініціалізувати жоден бекенд."
  );
};
```

Рисунок 3.22 – Код функції ініціалізації моделі з файлу Trainer.jsx

Для обробки кадрів на сторінці обов’язково має бути присутнім об’єкт з відео. У вебзастосунку він був залишений у структурі, але схований через стилі Tailwind CSS. Відтворення кадрів після обробки застосунком відбувається на полотні, що було створено під відео у структурі сторінки (рис. 3.23). Саму стилізацію описано у index.css (рис. 3.24).

```
<video
  ref={videoRef}
  id="video"
  width={WIDTH.toString()}
  height={HEIGHT.toString()}
  autoPlay
  muted
  playsInline
></video>
<canvas
  className="stream"
  ref={canvasRef}
  width={WIDTH.toString()}
  height={HEIGHT.toString()}
/>
```

Рисунок 3.23 – Код з реалізації відео та полотна у структурі сторінки

```
video {
  @apply absolute opacity-0 h-max;
}

.stream {
  @apply w-full border-l-4 border-b-4 border-amber-950;
}
```

Рисунок 3.24 – Частина коду з файлу index.css

Щоб скопіювати зображення з відео на полотно, здійснюється перевірка стану роботи камери. Якщо все працює, то перемальовуємо кадр з відео на полотно. Код зображено на рисунку 3.25.

```
if (video?.readyState < 2) {
  return requestAnimationFrame(detect);
}

// Малюємо кадр з відео
ctx.drawImage(video, 0, 0, canvas.width, canvas.height);
```

Рисунок 3.25 – Частина функції з обробки кадрів за допомогою моделі

Перейдемо до відстеження ключових точок на зображені. Спочатку передаємо заповнене полотно до встановленої моделі. З неї отримуємо сформовані ключові точки (рис. 3.26).

```
let poses;
if (sessionStatusRef.current) {
  poses = await detector.estimatePoses(canvas, {
    maxPoses: 1,
    flipHorizontal: false,
  });
}
```

Рисунок 3.26 – Код з формування ключових точок

Якщо вдалося знайти людину у кадрі, то фільтруємо дані за заданою мінімальною точністю 0,4 у нову колекцію Map (рис. 3.27).

```
if (poses.length > 0) {
  const minKeypointScore = 0.4;
  const keypointsMap = {};

  poses[0].keypoints.forEach((kp) => {
    if (kp.score >= minKeypointScore) {
      keypointsMap[kp.name] = kp;
    }
  });
```

Рисунок 3.27 – Код з фільтрування ключових точок

Після обробки полотна можемо перейти до малювання додаткових елементів. Спочатку перевіряються відповідні налаштування збережені у локальному сховищі. Якщо налаштування включено, то відбувається малювання елемента.

Одним з таких елементів є кількість кадрів вебзастосунку у секунду (рис. 3.28). Він потрібен для відстеження швидкості роботи. Розраховується значення кожну секунду. Запам’ятовується минулий час і поточний. Кожен кадр тимчасове значення загальної кількості кадрів збільшується на одиницю. Після спливання секунди береться загальна отримана кількість кадрів і ділиться на пройдений час у мілісекундах.

```
if (settingsRef.current.showFPS) {
  // Малюємо фон для FPS
  ctx.fillStyle = "black";
  ctx.fillRect(8, 10, 100, 25);

  // Показуємо FPS
  ctx.font = "20px Arial";
  ctx.fillStyle = "white";
  ctx.fillText(`FPS: ${avgFPSRef.current}`, 10, 30);
}
```

Рисунок 3.28 – Код з відображення кількості кадрів в секунду на полотні

Таким чином вдається отримати середнє арифметичне кількості кадрів в секунду (рис. 3.29).

```
const currentTime = performance.now();
framesCountRef.current += 1;
const elapsed = currentTime - lastAvgFPSUpdateTimeRef.current;
if (elapsed >= 1000) {
  avgFPSRef.current = Math.round(
    framesCountRef.current / (elapsed / 1000)
  );
  framesCountRef.current = 0;
  lastAvgFPSUpdateTimeRef.current = currentTime;
}
```

Рисунок 3.29 – Код з обчислення частоти кадрів в секунду

Наступним кроком розглянемо процес малювання ключових точок. Зі сформованої колекції витягуємо значення і на полотні за цими координатами зображуємо червоні кола (рис. 3.30).

```
if (settingsRef.current.showKeypoints) {
  // Зображуємо ключові точки
  ctx.fillStyle = "red";
  ctx.beginPath();
  Object.values(keypointsMap).forEach((kp) => {
    ctx.moveTo(kp.x, kp.y);
    ctx.arc(kp.x, kp.y, 5, 0, 2 * Math.PI);
  });
  ctx.fill();
}
```

Рисунок 3.30 – Код з малювання ключових точок

Тепер варто їх об’єднати зеленими лініями. Це можна зробити за допомогою перебору заданих масивом ключів у колекції. Беремо координати першої точки і проводимо лінію до другої (рис. 3.31).

У процесі розпізнавання та контролювання вправи формується інформація про стан сесії, ціль з тренування, підказки та поточна кількість виконаних повторів. Її відображення відбувається у бічній панелі (рис. 3.32).

Компонент з підказками обробляє передану строку та виводить їх у вигляді списку (рис. 3.33).

```
if (settingsRef.current.showSkeleton) {
  const skeletonConnections = [
    [bodyParts.LEFT_SHOULDER, bodyParts.RIGHT_SHOULDER],
    [bodyParts.LEFT_SHOULDER, bodyParts.LEFT_ELBOW],
    [bodyParts.LEFT_ELBOW, bodyParts.LEFT_WRIST],
    [bodyParts.RIGHT_SHOULDER, bodyParts.RIGHT_ELBOW],
    [bodyParts.RIGHT_ELBOW, bodyParts.RIGHT_WRIST],
    [bodyParts.LEFT_SHOULDER, bodyParts.LEFT_HIP],
    [bodyParts.RIGHT_SHOULDER, bodyParts.RIGHT_HIP],
    [bodyParts.LEFT_HIP, bodyParts.RIGHT_HIP],
    [bodyParts.LEFT_HIP, bodyParts.LEFT_KNEE],
    [bodyParts.LEFT_KNEE, bodyParts.LEFT_ANKLE],
    [bodyParts.RIGHT_HIP, bodyParts.RIGHT_KNEE],
    [bodyParts.RIGHT_KNEE, bodyParts.RIGHT_ANKLE],
  ];

  // Зображуємо скелет між ключовими точками
  ctx.strokeStyle = "lime";
  ctx.lineWidth = 2;
  ctx.beginPath();
  skeletonConnections.forEach(([partA, partB]) => {
    const kpA = keypointsMap[partA];
    const kpB = keypointsMap[partB];
    if (kpA && kpB) {
      ctx.moveTo(kpA.x, kpA.y);
      ctx.lineTo(kpB.x, kpB.y);
    }
  });
  ctx.stroke();
}
```

Рисунок 3.31 – Код з малювання ліній між точками

```
<div className="information">
  <h3 className="information-title">
    тренування {sessionStatus ? "активне✅" : "неактивне❌"}
  </h3>
  <p>Ціль: {goal}</p>
  <p>Виконано повторів: {repsCountRef.current}</p>
</div>
<Hints hints={hints} />
```

Рисунок 3.32 – Фрагмент коду з бокової панелі

```
const Hints = ({hints}) => {
  if (hints.length > 0) {
    return (
      <div className="hints">
        <h2>підказки</h2>
        <ul className="hints-list">
          {hints.split("\n").map((hint, index) => (
            <li key={index}>{hint}</li>
          ))}
        </ul>
      </div>
    );
  }
  return <></>;
};

export default Hints;
```

Рисунок 3.33 – Код компоненту Hints

Контролювання правильності виконання вправи відбувається в окремій функції з іншого файлу. У неї передаємо колекцію з точками, стан підказок, назву поточної вправи, стан виконання вправи, функції з відтворення звуків, стан кількості виконаних повторів та стан зауважень (рис. 3.34).

```
detectExercise(
  keypointsMap,
  setHints,
  exercise,
  exercisePositionRef,
  settingsRef.current.playSounds ? playExerciseEndSound : () => {},
  settingsRef.current.playSounds ? playExerciseGoalSound : () => {},
  repsCountRef,
  remarksRef,
  isNaN(Number(goal)) ? 0 : Number(goal)
);
```

Рисунок 3.34 – Код виклику функції з контролю виконання вправ

Після завершення виконання вправи і закриття тренування користувачем, буде відображатися звіт (рис. 3.35).

```
{isReportOpen && (
  <Report
    exercise={exercise}
    repsDone={repsCountRef.current}
    repsGoal={isNaN(Number(goal)) ? 0 : Number(goal)}
    remarks={remarksRef.current}
  />
)}
```

Рисунок 3.35 – Код для відображення компоненту Report

У звіті присутні акордеони. Усі вони мають анімацію відкриття та закриття їх вмісту (рис. 3.36). У першому акордеоні формується інформація про продуктивність. На основі виконаної кількості повторень та максимальної кількості визначається, яке повідомлення буде виведено користувачу (рис. 3.37).

```
if (content.style.maxHeight && content.style.maxHeight !== "0px") {
  content.style.maxHeight = "0px";
  icon.innerHTML = downSVG;
} else {
  content.style.maxHeight = content.scrollHeight + "px";
  icon.innerHTML = upSVG;
}
```

Рисунок 3.36 – Код для анімації відкриття та закриття акордеону

```
<div
  id="content-1"
  className="max-h-0 overflow-hidden transition-all duration-300 ease-in-out px-4 mb-2"
>
  <p>
    {repsDone > repsMax
      ? "✅ Встановлено новий рекорд!"
      : repsDone === repsMax
      ? "✅ Повторено старий рекорд!"
      : repsDone > repsGoal
      ? "✅ Ціль перевиконано!"
      : repsDone === repsGoal
      ? "✅ Ціль виконано!"
      : "❌ Не вдалося досягти цілі."}
  </p>
  <p>
    Виконано повторень: {repsDone}
    <br />
    Ціль: {repsGoal}
    <br />
    Максимум: {repsMax}
  </p>
</div>
```

Рисунок 3.37 – Код до першого акордеону у компоненті Report

Список зауважень відображається у другому акордеоні (рис. 3.38). Якщо зауваження відсутні, буде виведено відповідний текст.

```
<div
  id="content-2"
  className="max-h-0 overflow-hidden transition-all duration-300 ease-in-out px-4 mb-2"
>
  {Object.entries(remarks).length > 0
    ? Object.entries(remarks).map(([key, value]) => {
        return (
          <div className="flex flex-row justify-between">
            <div>❌ {key}</div>
            <div>{value + " раз(и/ів)"}</div>
          </div>
        );
      })
    : "✅ Відсутні!"}
</div>
```

Рисунок 3.38 – Код до другого акордеону з файлу Report.jsx

У звіті надається можливість збереження результату тренування чи відмови від нього. Якщо користувач обирає варіант зі збереженням, то викликається спеціальний алгоритм дій (рис. 3.39).

```
onClick={() => {
  if (
    exerciseProgress.stage === 0 ||
    exerciseProgress.stage === 7
  ) {
    exerciseProgress.max = repsDone;
  } else {
    exerciseProgress.max = Math.max(
      exerciseProgress.max,
      repsDone
    );
  }
  if (exerciseProgress.stage === 7) {
    exerciseProgress.stage = 1;
  } else {
    exerciseProgress.stage += 1;
  }
  localStorage.setItem(
    "exercisesProgress",
    JSON.stringify({
      ...parsed,
      [exercise]: {...exerciseProgress},
    })
  );
  dispatch(closeReport());
  dispatch(closeTrainer());
}}
```

Рисунок 3.39 – Код для збереження результатів тренування

Перейдемо до функції з контролювання виконання вправ. У ній присутні методи для додавання та видалення підказок (рис. 3.40). За допомогою них вдається у реальному часі вчасно інформувати користувача.

```
const DEFAULT_TIME = 4000;
const EXERCISE_TIME = 1000;

const addHint = (hint, time) => {
  setHints((prevHints) => {
    if (prevHints.includes(hint)) return prevHints;
    if (time === DEFAULT_TIME) {
      remarksRef.current[hint] = remarksRef.current[hint]
        ? remarksRef.current[hint] + 1
        : 1;
      setTimeout(() => {
        setHints((hintsAfterTimeout) => hintsAfterTimeout.replace(hint, ""));
      }, time);
    }
    return prevHints + hint;
  });
};

const removeHint = (hint) => {
  setHints((prevHints) => prevHints.replace(hint, ""));
};
```

Рисунок 3.40 – Код для додавання та видалення підказок з файлу logic.js

Для перевірки наявності у кадрі хоча б однієї зі сторін тіла людини використовуються логічні оператори (рис. 3.41).

```
const leftShoulder = getPoint(bodyParts.LEFT_SHOULDER);
const rightShoulder = getPoint(bodyParts.RIGHT_SHOULDER);
const leftElbow = getPoint(bodyParts.LEFT_ELBOW);
const rightElbow = getPoint(bodyParts.RIGHT_ELBOW);
const leftWrist = getPoint(bodyParts.LEFT_WRIST);
const rightWrist = getPoint(bodyParts.RIGHT_WRIST);
const leftHip = getPoint(bodyParts.LEFT_HIP);
const rightHip = getPoint(bodyParts.RIGHT_HIP);
const leftAnkle = getPoint(bodyParts.LEFT_ANKLE);
const rightAnkle = getPoint(bodyParts.RIGHT_ANKLE);
const leftKnee = getPoint(bodyParts.LEFT_KNEE);
const rightKnee = getPoint(bodyParts.RIGHT_KNEE);

const leftSide =
  leftWrist && leftElbow && leftShoulder && leftHip && leftAnkle && leftKnee;
const rightSide =
  rightWrist &&
  rightElbow &&
  rightShoulder &&
  rightHip &&
  rightAnkle &&
  rightKnee;

if (leftSide || rightSide) {
```

Рисунок 3.41 – Код для перевірки наявності сторони тіла людини у кадрі

Якщо було запущено тренування для відтискань, то будуть перевірятися відповідні умови виконання.

Спочатку здійснюється опис у методах самих вимог (рис. 3.42).

На наступному етапі формується функція для перевірки виконання вимог для однієї сторони (рис. 3.43).

Потім відбуваються перевірки з наявності сторін у кадрі та їх опрацювання (рис. 3.44).

Наприкінці здійснюється фінальна перевірка на стадію виконання вправи (рис. 3.45).

Для присідань процес контролю працює за аналогічною схемою, але з іншими перевірками. Це можна побачити на рисунках 3.46 та 3.47.

```
const isWristUnderShoulder = (shoulderAngle) => {
  return shoulderAngle <= 90;
};

const areElbowsAnglesNormal = () => {
  return elbowsYMax < wristsYMax;
};

const areWristsOnFloor = () => {
  return avgWrists.y > avgHips.y;
};

const areLegsNarrowEnough = () => {
  return anklesYMax <= wristsYMax;
};

const isBackStraight = (shoulder, hip) => {
  const dx = shoulder.x - hip.x;
  const dy = shoulder.y - hip.y;
  const angleRad = Math.atan2(dx, dy);
  const angleFromVertical = 180 - Math.abs(angleRad * (180 / Math.PI));
  return angleFromVertical <= 100;
};
```

Рисунок 3.42 – Код методів для перевірки вимог виконання з відтискань

```
const processSide = (hip, shoulder, wrist, elbow, ankle) => {
  avgHips = hip;
  avgWrists = wrist;
  elbowsYMax = Math.max(elbowsYMax, elbow.y);
  anklesYMax = Math.max(anklesYMax, ankle.y);
  wristsYMax = Math.max(wristsYMax, wrist.y);

  const shoulderAngle = calculateAngle(elbow, shoulder, hip);
  if (!isWristUnderShoulder(shoulderAngle)) {
    addHint("Розташуйте долоні на ширині плечей\n", DEFAULT_TIME);
  }

  if (!isBackStraight(shoulder, hip)) {
    addHint("Тримайте таз нижче плечей\n", DEFAULT_TIME);
  }

  const elbowAngle = calculateAngle(wrist, elbow, shoulder);
  avgElbowAngle += elbowAngle;
};
```

Рисунок 3.43 – Код для опрацювання сторони тіла людини

```
if (leftSide) {
  processSide(leftHip, leftShoulder, leftWrist, leftElbow, leftAnkle);
}
if (rightSide) {
  processSide(
    rightHip,
    rightShoulder,
    rightWrist,
    rightElbow,
    rightAnkle
  );
}
if (leftSide && rightSide) {
  avgElbowAngle /= 2;
  avgWrists = toAvg(leftWrist, rightWrist);
  avgHips = toAvg(leftHip, rightHip);

  if (!areLegsNarrowEnough()) {
    addHint("Розташуйте ноги на ширині плечей\n", DEFAULT_TIME);
  }
}

if (!areElbowsAnglesNormal()) {
  addHint(
    "Згинайте лікті під кутом 45 градусів до тулуба\n",
    DEFAULT_TIME
  );
}
```

Рисунок 3.44 – Код для перевірки виконання вимог з виконання вправи

```
if (areWristsOnFloor()) {
  if (avgElbowAngle <= 100) {
    setExercisePosition("end");
  } else if (avgElbowAngle >= 160) {
    setExercisePosition("begin");
  }
} else {
  addHint("Долоні мають бути на підлозі\n", DEFAULT_TIME);
}
```

Рисунок 3.45 – Код з фінальною перевіркою для встановлення стадії виконання відтискань

```
const isBackStraight = (shoulder, hip) => {
  const dx = shoulder.x - hip.x;
  const dy = shoulder.y - hip.y;
  const angleRad = Math.atan2(dx, dy);
  const angleFromVertical = 180 - Math.abs(angleRad * (180 / Math.PI));
  return angleFromVertical <= 50;
};

const areKneesBehindToes = (knee, ankle, hipKneeLength) => {
  const kneeAnkleLengthX = Math.abs(knee.x - ankle.x);
  return kneeAnkleLengthX <= hipKneeLength * 0.5;
};

const processSide = (shoulder, hip, knee, ankle) => {
  const hipKneeLength = vectorLength(toVector(hip, knee));

  if (!isBackStraight(shoulder, hip)) {
    addHint("Тримайте спину рівно\n", DEFAULT_TIME);
  }

  if (!areKneesBehindToes(knee, ankle, hipKneeLength)) {
    addHint(
      "Коліна не повинні виходити за межі носків чи знаходитися за п'ятами\n",
      DEFAULT_TIME
    );
  }

  avgKneeAngle += calculateAngle(hip, knee, ankle);
};
```

Рисунок 3.46 – Перша частина коду з контролювання виконання присідань

```
if (leftSide) {
  processSide(leftShoulder, leftHip, leftKnee, leftAnkle);
}
if (rightSide) {
  processSide(rightShoulder, rightHip, rightKnee, rightAnkle);
}
if (leftSide && rightSide) {
  avgKneeAngle /= 2;
}
if (avgKneeAngle <= 90) {
  setExercisePosition("end");
} else if (avgKneeAngle >= 160) {
  setExercisePosition("begin");
}
```

Рисунок 3.47 – Друга частина коду з контролювання виконання присідань

Наведемо код для скручувань на прес на рисунках 3.48 та 3.49.

```
const processSide = (shoulder, hip, knee, ankle) => {
  avgKneeAngle += calculateAngle(hip, knee, ankle);
  avgShoulders = shoulder;
  avgHips = hip;
  avgKnees = knee;
};

if (leftSide) {
  processSide(leftShoulder, leftHip, leftKnee, leftAnkle);
}
if (rightSide) {
  processSide(rightShoulder, rightHip, rightKnee, rightAnkle);
}
if (leftSide && rightSide) {
  avgKneeAngle /= 2;
  avgShoulders = toAvg(leftShoulder, rightShoulder);
  avgHips = toAvg(leftHip, rightHip);
  avgKnees = toAvg(leftKnee, rightKnee);
}
```

Рисунок 3.48 – Перша частина коду з контролювання виконання скручувань

```
const getBackVerticalAngle = (shoulders, hips) => {
  const dx = shoulders.x - hips.x;
  const dy = shoulders.y - hips.y;
  const angleRad = Math.atan2(dx, dy);
  const angleFromVertical = Math.abs(angleRad * (180 / Math.PI));
  return angleFromVertical;
};

const verticalAngle = getBackVerticalAngle(avgShoulders, avgHips);

if (avgKnees.y < avgShoulders.y) {
  if (avgKneeAngle > 90) {
    addHint("Кут в колінах повинен бути до 90° включно\n", DEFAULT_TIME);
  }

  if (verticalAngle <= 90) {
    setExercisePosition("begin");
  } else if (verticalAngle >= 100) {
    setExercisePosition("end");
  }
}
```

Рисунок 3.49 – Друга частина коду з контролювання виконання скручувань

Підказки з загального виконання вправ формуються окремо (рис. 3.50). Методи з підрахунку кількості виконаних повторень, зміни стадії виконання вправи та отримання поточної стадії наведено на рисунку 3.51.

```
const beginHint =
  exercise !== exercises.CRUNCHES
    ? "Прийміть кінцеве положення\nВдихайте повітря під час виконання\n"
    : "Прийміть кінцеве положення\nВидихайте повітря під час виконання\n";
const endHint =
  exercise !== exercises.CRUNCHES
    ? "Прийміть початкове положення\nВидихайте повітря під час виконання\n"
    : "Прийміть початкове положення\nВдихайте повітря під час виконання\n";

const currentExercisePosition = getExercisePosition();
if (currentExercisePosition === "end") {
  removeHint(beginHint);
  addHint(endHint, EXERCISE_TIME);
} else if (currentExercisePosition === "begin") {
  removeHint(endHint);
  addHint(beginHint, EXERCISE_TIME);
}
```

Рисунок 3.50 – Код з додавання загальних підказок для виконання вправ

```
const setExercisePosition = (newState) => {
  if (exercisePositionRef.current !== newState) {
    exercisePositionRef.current = newState;
    if (newState === "end") {
      repsCountRef.current += 1;
      if (repsCountRef.current < repsGoal) {
        playExerciseEndSound();
      } else {
        playExerciseGoalSound();
      }
    }
  }
};

const getExercisePosition = () => {
  return exercisePositionRef.current;
};
```

Рисунок 3.51 – Код з методами для зміни та отримання стадії виконання вправи

Таким чином вдалося створити робочий вебзастосунок з функціональним та приємним інтерфейсом.

3.3 Тестування розробленого вебзастосунку та аналіз результатів

При першому запуску застосунку (рис. 3.52) бачимо головну сторінку сайту. На ній видно назву застосунку та його слоган.

Рисунок 3.52 – Головна сторінка застосунку

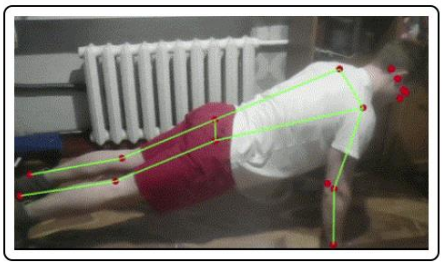
Нижче можемо побачити список вправ з можливістю запуску тренування, перегляду інструкції та скидання прогресу для кожної з них окремо. Наведено шкалу з максимальною кількістю виконаних повторень за цикл, відсотком виконання циклу та анімованою доріжкою. Після виконання першого тренування зліва має з’явитися максимум користувача. Відсоток циклу також має змінити свою цифру, а доріжка на задньому фоні почне збільшувати свій розмір у правому напрямку.

Спробуємо переглянути інструкції до кожної вправи окремо. У них мають бути описані правила виконання вправ та поруч наведено анімоване зображення з правильною технікою виконання у ролі прикладу. Вигляд інструкцій можна побачити на рисунках 3.53 – 3.55.

ІНСТРУКЦІЯ (ВІДТИСКАННЯ)

Техніка виконання:
– початкова позиція:
1) тіло займає пряму лінію від голови до п'ят. Долоні розташовані на ширині плечей або трохи ширше;
2) пальці ніг знаходяться на підлозі, ноги разом або трохи розведені;
3) руки повинні бути під плечима, пальці рук дивляться вперед;
– техніка руху:
1) опускати тіло, згинаючи лікті під кутом 45 градусів до тулуба;
2) грудні м'язи мають опуститися якомога ближче до підлоги, але не торкатися її;
3) лікті не повинні виходити надто широко в сторони, щоб уникнути навантаження на плечі;
4) тіло залишається в одній лінії (поперек не прогинається, сідниці не піднімаються вгору);
– дихання:
1) вдих при опусканні;
2) видих при підйомі;
– типові помилки:
1) неправильна постановка рук (занадто вузько або широко);
2) прогин попереку або підняття тазу вгору;
3) недостатня амплітуда руху.

Приклад виконання вправи:



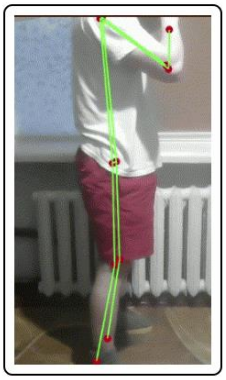
OK

Рисунок 3.53 – Інструкція з виконання відтискань

ІНСТРУКЦІЯ (ПРИСІДАННЯ)

Техніка виконання:
– початкова позиція:
1) ноги на ширині плечей або трохи ширше, стопи розвернуті назовні приблизно на 15-30 градусів;
2) спина повинна бути прямою, плечі злегка опущені назад, голова дивиться прямо перед собою;
– техніка руху:
1) починати рух з відведення тазу назад, ніби намагаємось сісти на невидимий стілець;
2) коліна повинні рухатися у напрямку до носків, але не виходити за їх межі;
3) стегна опускаються до рівня паралелі з підлогою або нижче, залежно від гнучкості людини;
4) вага тіла повинна бути на п'ятах, а не на носках;
– дихання:
1) вдих при опусканні;
2) видих при підйомі;
– типові помилки:
1) округлення спини під час виконання вправи;
2) занадто сильний нахил тулуба вперед;
3) перенесення ваги на носки, що може спричинити перенавантаження колін.

Приклад виконання вправи:



OK

Рисунок 3.54 – Інструкція з виконання присідань

Рисунок 3.55 – Інструкція з виконання скручувань

Спробуємо відкрити тренування з відтискань. При першому відкритті має з'явитися повідомлення з проханням надання доступу до камери користувача (рис. 3.56).

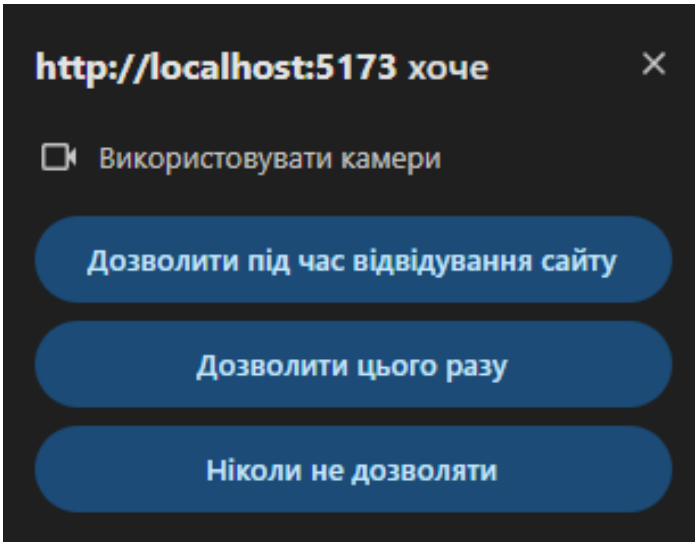


Рисунок 3.56 – Повідомлення про прохання надання доступу до камери

Якщо користувач не надасть доступ, то відкриється модальне вікно з повідомленням про це (рис. 3.57).

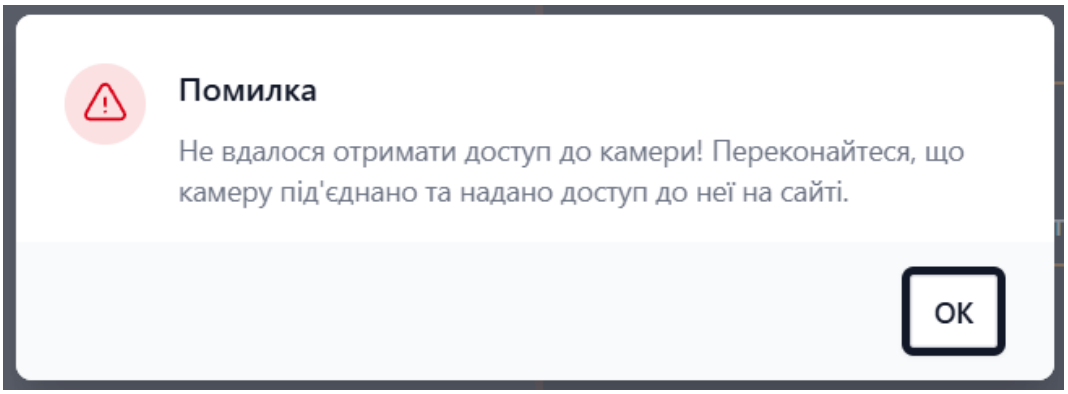


Рисунок 3.57 – Модальне вікно з повідомленням про проблеми з камерою

За відсутності проблем з камерою та вебзастосунком має відкритися саме тренування (рис. 3.58).

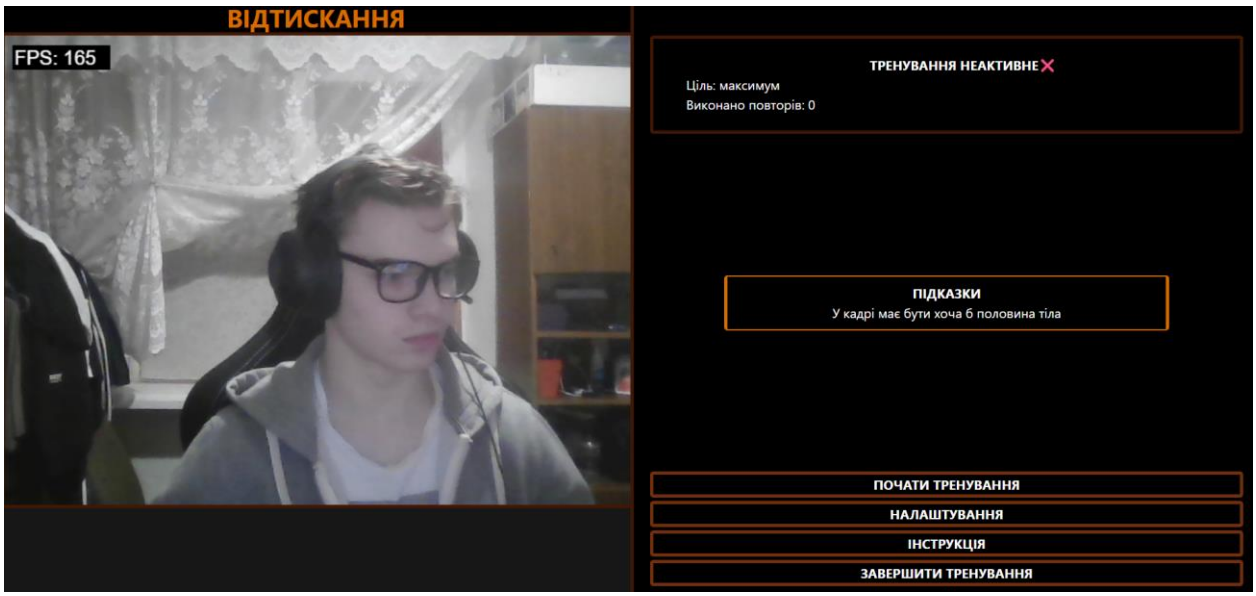


Рисунок 3.58 – Інтерфейс тренування

Зверху у лівій частині можна побачити назву вправи для якої було відкрито тренування. Під назвою користувач побачить полотно, що транслює кадри з камери. Праворуч відображається стан тренування, поточна ціль і кількість вже виконаних повторень. Нижче формуються підказки.

У правій нижній частині сторінки присутнє меню для контролювання процесу тренування. Розглянемо, що робить кожна кнопка з цього списку.

Якщо перейти до налаштувань, то можна побачити кілька пунктів (рис. 3.59):

– показувати FPS (контролює стан відображення кількості кадрів в секунду на полотні);

– показувати ключові точки (контролює стан відображення ключових точок на полотні);

– показувати скелет (контролює стан відображення ліній між ключовими точками);

– програвати звуки (контролює стан програвання звуків під час виконання повторень).

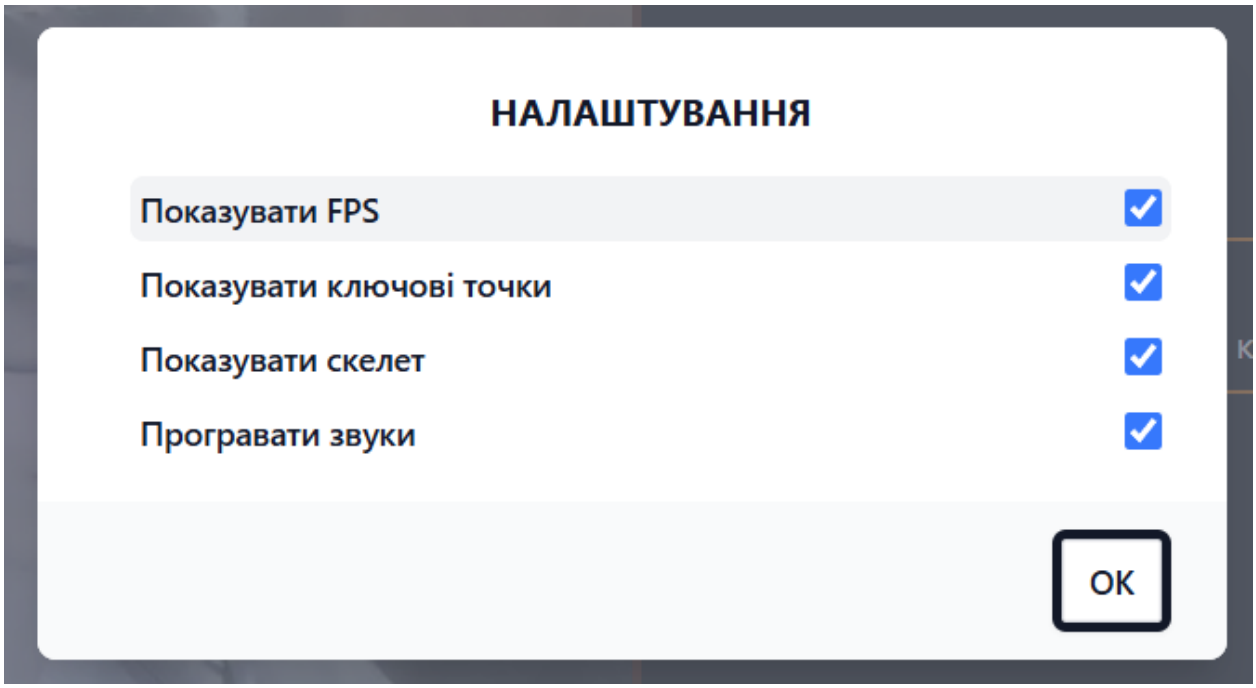


Рисунок 3.59 – Вікно з налаштуваннями

При наведенні на пункт з налаштувань, він пофарбує свій фон у затемнений. Якщо навести на прапорець, то він збільшить свій розмір. Для контролювання станом налаштування необхідно лише натиснути на прапорець біля потрібного пункту.

Якщо користувач натисне у тренуванні на кнопку з відкриття інструкції, то відкриється таке саме вікно з інструкцією до вправи як і у головному меню вебзастосунку.

Якщо почати тренування, то стан сесії зміниться на активний. На полотні почнуть відображатися ключові точки та лінії між ними, якщо це включено у налаштуваннях. Підказки почнуть формуватися у реальному часі. Якщо вправу буде виконано правильно, то буде програватися звук та зараховуватиметься повторення (рис. 3.60).

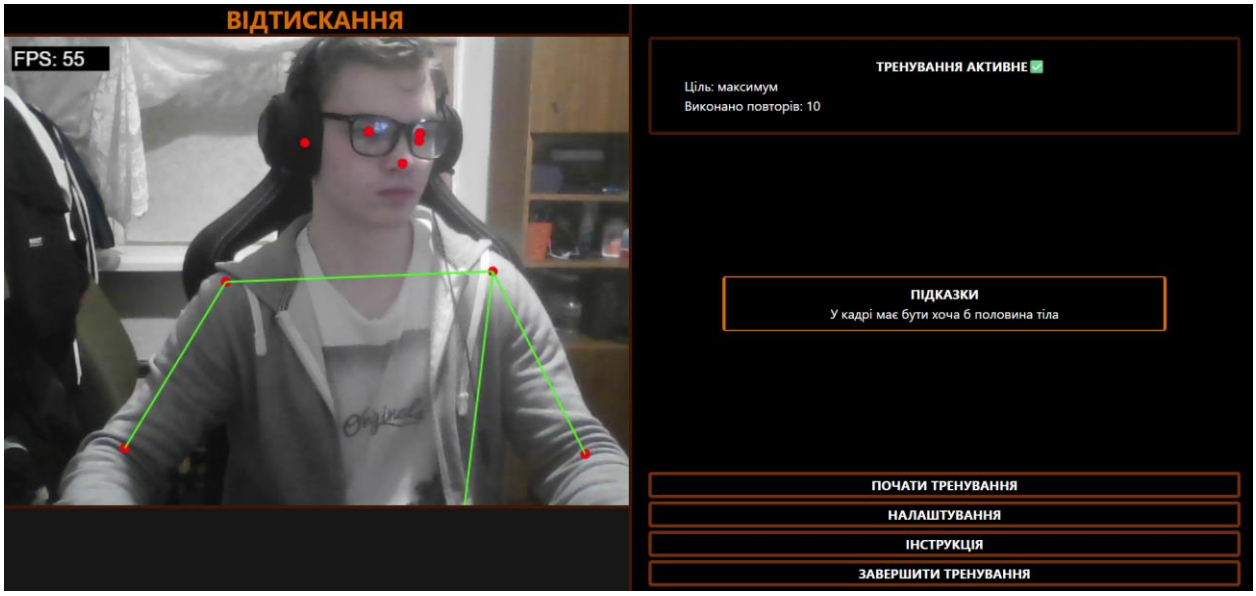


Рисунок 3.60 – Зміна інтерфейсу після початку тренування
та виконання 10 повторень

Після виконання тренування його можна завершити. Перед користувачем з’явиться звіт (рис. 3.61). У продуктивності буде порівняно результати з ціллю та максимумом. Нижче буде сформовано зауваження щодо виконання та їх кількість. Якщо зауваження відсутні, то про це буде написано. Користувач обирає бажає він зберегти результати тренування чи ні. Якщо не зберігати, то прогрес з вправи у головному меню не зазнає ніяких змін. Збережемо результат для перегляду змін. Тепер у вправі з відтискань оновився максимум і його видно на шкалі.

Рисунок 3.61 – Звіт з виконання вправи

Відсоток з виконання циклу збільшився, як і очікувалося. Зміни з прогресу на шкалі у головному меню можна переглянути на рисунку 3.62.

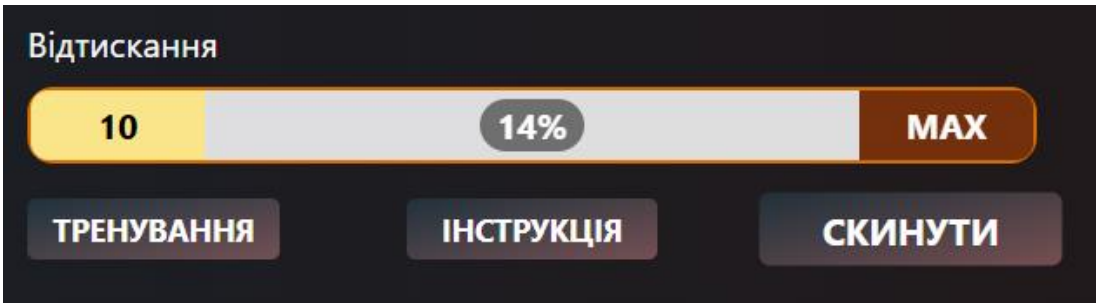


Рисунок 3.62 – Оновлений прогрес з відтискань

Переглянемо як система буде формувати подальші цілі для тренувань у циклі. Очікуються такі цілі: 6, 8, 7, 9, 7, 9, максимум. Переглянути сформовані застосунком цілі можна на рисунках 3.63 – 3.69.

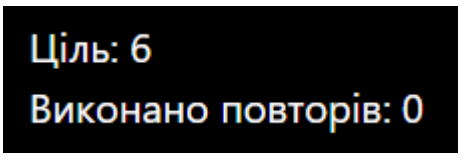


Рисунок 3.63 – Ціль другого тренування

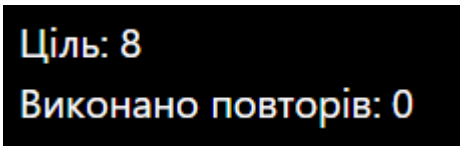


Рисунок 3.64 – Ціль третього тренування

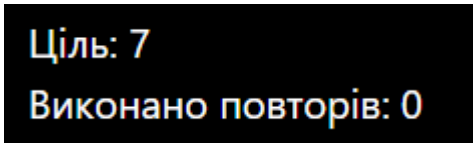


Рисунок 3.65 – Ціль четвертого тренування

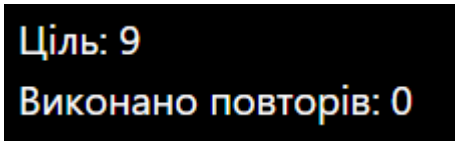


Рисунок 3.66 – Ціль п’ятого тренування

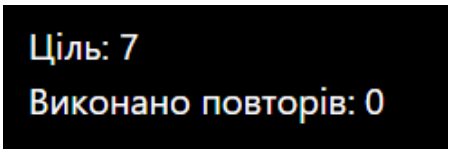


Рисунок 3.67 – Ціль шостого тренування

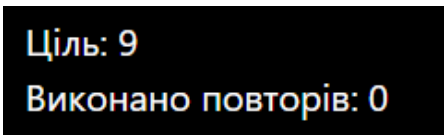


Рисунок 3.68 – Ціль сьомого тренування

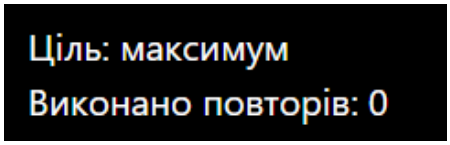


Рисунок 3.69 – Ціль восьмого тренування

Як бачимо за рисунками вебзастосунок правильно формує цілі. Спробуємо інший сценарій використання. Якщо користувач на другому тренуванні зробить більше повторень, ніж було максимально зроблено у цілому (рис. 3.70). У шкалі повинно оновитися значення максимуму та ціль третього тренування має становити 15. Результат таких маніпуляцій зображено на рисунках 3.71 та 3.72.

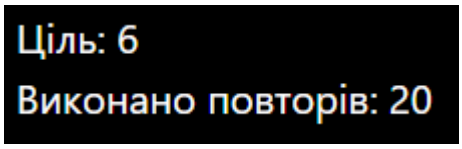


Рисунок 3.70 – Ціль другого тренування та оновлений рекорд

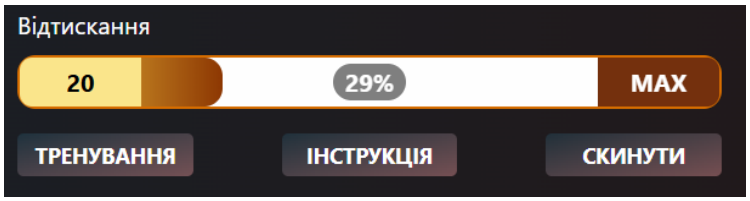


Рисунок 3.71 – Оновлена шкала з відтискань

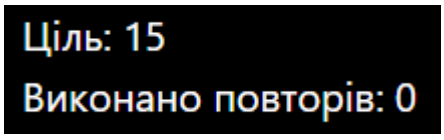


Рисунок 3.72 – Оновлена ціль третього тренування

З цим сценарієм вебзастосунок успішно впорався. Перейдемо до іншого. Якщо користувач зрозуміє, що не виконує план тренувань з циклу чи хоче просто почати цикл з початку, то є можливість скидання прогресу з вправи до початкового стану. Перевіримо цей сценарій (рис. 3.73).

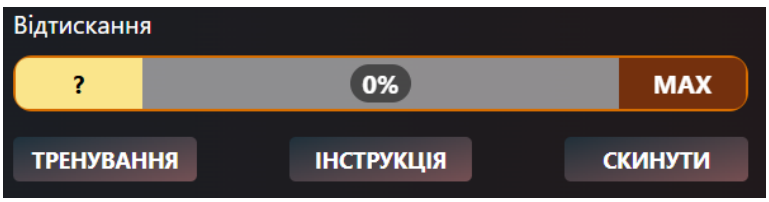


Рисунок 3.73 – Оновлена шкала відтискань після скидання

З результатів тестування зрозуміло, що застосунок успішно справляється з усіма потрібними завданнями. Результати тестування задовольняють усім поставленим вимогам на початку розробки вебзастосунку. Програма працює без критичних збоїв.

3.4 Перспективи подальшої розробки

Розроблений вебзастосунок чудово виконує усі поставлені задачі у заданих обмеженнях. Ручне тестування показало задовільні результати роботи застосунку. Серед слабких місць програми найскладнішим є контроль правильності виконання вправ. Це відбувається за декількох причин.

Причини:

− швидкодія моделі на слабких пристроях;

− якість камери пристрою;

− складність моделі;

− відсутність підтримки браузерами нових обчислювальних рушіїв;

− недостатньо оптимізований застосунок.

Над усіма цими недоліками необхідно додатково працювати. Графічний інтерфейс теж може бути краще оптимізовано для зменшення ресурсних витрат та затримок у роботі. Додатково до технічних обмежень накладається потреба у формуванні основних вимог до виконання вправ.

Частина з вимог не є універсальною і занадто складна в обчисленнях, що створює потребу у проведенні додаткових досліджень і пошуках оптимальніших способів. Альтернативним шляхом може бути створення конструктора вправ для надання можливості користувачам самим визначати і формувати вимоги до вправ. Такий шлях буде потребувати ширших знань від користувачів і може спричинити додаткову складність, тому його не було реалізовано. Натомість можна додати додаткові налаштування зі зміни допустимих кутів при виконанні вправи у деяких умовах. Таке рішення зможе додати універсальності застосунку.

Розширення списку вправ може дозволити вийти на ширшу аудиторію користувачів. У подальшому можливо додати вправи для виконання у спортзалі чи на подвір’ї, тим самим уникнути специфічності застосунку. Ще варто спробувати розширити список положень тіла людини для виконання вправ.

Для тренованих користувачів є потреба у появі складнішої системи тренувань. Цикл на основі тренувань з одним підходом не є достатньо ефективним для них. Можливо додати навіть декілька програм для тренування на вибір користувачу з розподілом за рівнем складності та ціллю. Якщо змінити побудову застосунку, то можна створити програми, що будуть складатися з виконання декількох вправ за тренування.

Відстеження днів тренування користувача дозволить додати аналітику. Доречно було б додати розділ з ширшою аналітикою тренувань у якій були б присутні різні графіки для відстеження прогресу користувача.

Якщо виникне потреба для відстеження прогресу декількох користувачів за одним пристроєм, то варто додати систему реєстрації користувачів. Введення облікових записів створить можливість для зберігання даних користувачів на сервері. Без зберігання даних на сервері є альтернативний шлях у наданні можливості користувачу формування файлу з усією інформацію про його прогрес і можливості зчитування вебзастосунком таких файлів для завантаження збереженого прогресу.

Таким чином існує велика кількість перспективних ідей щодо покращення вебзастосунку. Кожна ідея потребує додаткових досліджень, часу та уваги. Втілення у життя цих ідей дасть можливість виведення цього продукту на інформаційний ринок.

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і реалізований вебзастосунок для розпізнавання та контролювання правильності виконання фізичних вправ у реальному часі.

Було виконано такі поставлені задачі:

− проаналізовано сучасні програми та сайти спортивного напрямку з можливістю контролювання якості виконання фізичних вправ;

− проведено аналіз існуючих методів розпізнавання на базі CNN, порівняно та обрано оптимальну модель;

− розроблено повноцінний вебзастосунок на базі обраної моделі.

Було проаналізовано такі застосунки: Home Workout, Fit AI, Personal Fitness Trainer AI, FormCheck, Agit. Оптимальною моделлю для застосунку під задані вимоги між сучасних методів розпізнавання виявилася нейронна мережа MoveNet SinglePose Thunder У якості мови програмування для розробки було обрано JavaScript. Серед використаних фреймворків та бібліотек були React, Redux, TensorFlow.js, Tailwind CSS. Додатково було змодельовано структуру вебзастосунку засобами програми Rational Rose. Описано та побудовано діаграму варіантів використання та створено діаграму класів. Детально розглянуто етапи програмної реалізації. Після розробки було додатково проведено ручне тестування застосунку з різними можливими сценаріями його застосування. Дефектів під час тестування виявлено не було.

Як показав аналіз досліджуваних джерел в області інформаційних технологій на сьогоднішній день, ця сфера стрімко розвивається. Існує багато цікавих напрямків, ознайомлення з якими дозволить отримати глибше та ширше розуміння сфери комп’ютерних наук [31–50].

Результати роботи апробовано у вигляді тез доповіді під час Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» [51].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Home Workout – No Equipments. URL: https://apps.apple.com/us/app/home-workout-no-equipments/id1313192037 (дата звернення 10.04.2025).

2. FitAI. URL: https://fitaiapp.com/ (дата звернення 10.04.2025).

3. Personal Fitness Trainer AI. URL: https://github.com/chrisprasanna/Exercise_Recognition_AI (дата звернення 11.04.2025).

4. FormCheck. URL: https://apps.apple.com/ru/app/formcheck/id1563633996 (дата звернення 12.04.2025).

5. Agit. URL: https://www.agitapp.com/ (дата звернення 12.04.2025).

6. Вишнівський, Д. В. (2023). Модель та метод розпізнавання руху людини в режимі реального часу: кваліфікаційна робота другого (магістерського) рівня вищої освіти: 122 Комп'ютерні науки. Харків, 2023. 52 с.

7. Іванов, С., & Музичук, А. (2021). Про розпізнавання окремих ознак частин тіла людини з використанням обмежених обчислювальних ресурсів. *Вісник Львівського університету. Серія прикладна математика та інформатика*, (29).

8. Далявський, В., & Фечан, А. (2022). СИСТЕМИ РОЗПІЗНАВАННЯ ЛЮДЕЙ ЗА ДОПОМОГОЮ БІБЛІОТЕКИ YOLO V3. *Grail of Science*, (14-15), 331-339.

9. Гадевич, В. Ю. (2023). *Програмний модуль розпізнавання пози людини для системи комп'ютерного зору* (Doctoral dissertation, Тернопіль, ЗУНУ).

10. Трубіцин, О. О., & Аврунін, О. Г. (2022). Підхід до розробки модулю телемедичної системи автоматизованого аналізу рухів людини.

11. Кулигін, А. С. (2022). Дослідження моделей та методів комп’ютерного зору для відстеження рухів людського тіла: кваліфікаційна робота другого (магістерського) рівня вищої освіти: 121 Інженерія програмного забезпечення. Харків, 2022. 68 с.

12. Швець, О. С. (2024). Комп’ютерна система оптимізації фізичних навантажень спортсменів: кваліфікаційна робота першого (бакалаврського) рівня вищої освіти: 121 Інженерія програмного забезпечення. Запоріжжя, 2024. 69 с.

13. Потапенко, В. С. (2022). Нейронні мережі для розпізнавання об’єктів: кваліфікаційна робота другого (магістерського) рівня вищої освіти: 122 Комп’ютерні науки. Київ, 2022. 88 с.

14. Семенюта, І. Г. (2021). Розпізнавання зображень за допомогою нейронних мереж з використанням бібліотеки Tensor Flow: об’єктів: кваліфікаційна робота другого (магістерського) рівня вищої освіти: 122 Комп’ютерні науки. Полтава, 2021. 88 с.

15. Шевченко, О. Т. (2022). Методи обробки зображень на базі згорткових нейронних мереж: кваліфікаційна робота другого (магістерського) рівня вищої освіти: 123 Комп’ютерна інженерія. Харків, 2022. 62 с.

16. Waheed, S. R., Suaib, N. M., Rahim, M. S. M., Adnan, M. M., & Salim, A. A. (2021, April). Deep learning algorithms-based object detection and localization revisited. In *journal of physics: conference series* (Vol. 1892, No. 1, p. 012001). IOP Publishing.

17. Abbasi, S., & Rezaeian, M. (2021). Visual object tracking using similarity transformation and adaptive optical flow. *Multimedia Tools and Applications*, *80*(24), 33455-33473.

18. Josyula, R., & Ostadabbas, S. (2021). A review on human pose estimation. *arXiv preprint arXiv:2110.06877*.

19. Kansagara, J. A. (2022). Building an Efficient System to Catch Miscreants using Various Face Detection Techniques.

20. Sun, Y., Sun, Z., & Chen, W. (2024). The evolution of object detection methods. *Engineering Applications of Artificial Intelligence*, *133*, 108458.

21. Samkari, E., Arif, M., Alghamdi, M., & Al Ghamdi, M. A. (2023). Human pose estimation using deep learning: A systematic literature review. *Machine Learning and Knowledge Extraction*, *5*(4), 1612-1659.

22. Chung, J. L., Ong, L. Y., & Leow, M. C. (2022). Comparative analysis of skeleton-based human pose estimation. *Future Internet*, *14*(12), 380.

23. Singh, G., George, R. P., Ahmad, N., Hussain, S., Ather, D., & Kler, R. (2024). A deep learning approach for evaluating the efficacy and accuracy of PoseNet for posture detection. *International Journal of System Assurance Engineering and Management*, 1-10.

24. Mroz, S., Baddour, N., McGuirk, C., Juneau, P., Tu, A., Cheung, K., & Lemaire, E. (2021, December). Comparing the quality of human pose estimation with blazepose or openpose. In *2021 4th International Conference on Bio-Engineering for Smart Technologies (BioSMART)* (pp. 1-4). IEEE.

25. Kaushik, P., Lohani, B. P., Thakur, A., Gupta, A., Khan, A. K., & Kumar, A. (2023, September). Body posture detection and comparison between OpenPose, MoveNet and PoseNet. In *2023 6th International Conference on Contemporary Computing and Informatics (IC3I)* (Vol. 6, pp. 234-238). IEEE.

26. Dean, J. (2025). *Web Programming with HTML, CSS, and JavaScript*. Jones & Bartlett Learning.

27. Gerchev, I. (2022). *Tailwind CSS*. SitePoint Pty Ltd.

28. Nguyen, T. A. (2024). A comparative analysis of Webpack and Vite as build tools for JavaScript.

29. Khati Chhetri, A. (2024). Developing a Front-end web app using React.

30. Naik, P. G., & Oza, K. (2023). Awesome React. js (Unleash the Power of Modern UI Building). *International Institute of Organized Research (I2OR)*.

31. Tran, K. (2023). State management in react.

32. Agarwal, A., & Sharma, A. (2021). Web Application using Tensorflow.JS.

33. Laborde, G. (2021). *Learning TensorFlow. js*. "O'Reilly Media, Inc.".

34. Новожилова, М. В., & Стешенко, В. Ю. (2024). ПЕРЕВАГИ ВИКОРИСТАННЯ REDUX У ПОРІВНЯННІ З ТРАДИЦІЙНИМ REACT ПІД ЧАС РОЗРОБКИ ВЕБ-ІНТЕРФЕЙСУ. *Збірник наукових праць Харківського національного університету Повітряних Сил*, (3 (81)), 58-63.

35. Gerard, C. (2021). Practical Machine Learning in JavaScript. *Apress, Berkeley, CA*, 01-01.

36. Tvoroshenko I., and Gorokhovatskyi V. (2022) The Application of Hybrid Intelligence Systems for Dynamic Data Analysis, *International Journal of Engineering and Information Systems*, 6(2), pp. 40-48.

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

38. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2024) Improving the effectiveness of image classification structural methods by compressing the description according to the information content criterion, *Computers, Materials & Continua*, vol. 80, no. 2, pp. 3085-3106.

39. Gorokhovatskyi V., Tvoroshenko I., and Yakovleva O. (2024) Transforming image descriptions as a set of descriptors to construct classification features, *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 33, no. 1, pp. 113-125.

40. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Handwritten character recognition models based on convolutional neural networks, *International Journal of Academic Engineering Research*, 7(9), pp. 64-72.

41. Гороховатський В., Передрій О., Творошенко І., Марков Т. (2023) Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень, *Сучасні інформаційні системи*, 7(1), С. 5-13.

42. Gorokhovatskyi, V., Tvoroshenko, I., Kobylin, O., & Vlasenko, N. (2023). Search for visual objects by request in the form of a cluster representation for the structural image description, *Advances in Electrical and Electronic Engineering*, 21(1), pp. 19-27.

43. Gorokhovatskyi V., Chmutov Y., Tvoroshenko I., and Kobylin O. (2025) Reducing computational costs by compressing the structural description in image classification methods, *Advanced Information Systems*, vol. 9, no. 1, pp. 5–12.

44. Tvoroshenko I., Gorokhovatskyi V., Kobylin O., and Tvoroshenko A. (2023) Application of deep learning methods for recognizing and classifying culinary dishes in images, *International Journal of Academic and Applied Research*, 7(9), pp. 57-70.

45. Tvoroshenko I., Pomazan V., Gorokhovatskyi V., and Kobylin O. (2023) Application of video data classification models using convolutional neural networks, *International Journal of Academic and Applied Research*, 7(11), pp. 134-145.

46. Gorokhovatskyi V., Tvoroshenko I. (2023) Identification of visual objects by the search request. *International scientific symposium «INTELLIGENT SOLUTIONS-S». Computational intelligence (results, problems and perspectives). Decision making theory: proceedings of the international symposium, September 28, 2023, Kyiv-Uzhorod, Ukraine*, pp. 25-27.

47. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., Gadetska S., and Al-Dhaifallah M. (2023) Statistical data analysis models for determining the relevance of structural image descriptions, *IEEE Access*, vol. 11, pp. 126938-126949.

48. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., Hudáková M., and Gorokhovatskyi O. (2024) Application a committee of Kohonen neural networks to training of image classifier based on description of descriptors set, *IEEE Access*, vol. 12, pp. 73376-73385.

49. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., and Hudáková M. (2025) Image description compression in classification structural methods, *IEEE Access*, vol. 13, pp. 43631-43641.

50. Yakovleva O., Matúšová S., Tvoroshenko I., and Isaiev Y. (2024) Visitor counting based on video stream analysis from surveillance cameras to solve various business problems, *Verejná správa a regionálny rozvoj ekonómia, manažment a marketing*, XX(1), pp. 67-87.

51. Голубович Є. Д. (2025) Особливості існуючих вебзастосунків для розпізнавання та контролювання правильності виконання фізичних вправ. *Радіоелектроніка і молодь у XXI столітті: тези доповідей 29-го Міжнародного молодіжного форуму (Харків, 16–19 квітня 2025 р.)*. Харків: ХНУРЕ, 2025. Т.7. С. 38-40.