Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ АНАЛІЗУ ТРЕНУВАНЬ СПОРТСМЕНІВ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ DOCKER
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-21-3
Деркач І.В.
(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. Кобилін О. А.
(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 Деркачу Івану Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка вебзастосунку для аналізу тренувань спортсменів з використанням технології Docker

затверджена наказом університету від 19 травня 2025 року № 381Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 13 червня 2025 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, документація з веброзробки, офіційна документація Flask, Docker, SQLite, Python, бібліотеки аналізу даних Pandas та Matplotlib, ресурси інтернет-мережі.
4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд існуючих підходів до організації обліку тренувальної активності спортсменів та аналіз функціональних можливостей сучасних фітнес-застосунків.
2. Обґрунтування вибору технологій для реалізації вебзастосунку: Python, Flask, SQLite, Docker.
3. Розробка математичної моделі та алгоритмів обробки та аналізу даних тренувань.
4. Вибір методів візуалізації статистичних даних тренувального процесу.
5. Реалізація модуля розпізнавання техніки виконання вправ з використанням методів комп'ютерного зору.
6. Розробка програмної частини вебзастосунку з інтеграцією бази даних, системи візуалізації та контейнеризації.
7. Тестування роботи системи та побудова прикладів візуалізації даних.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) схема архітектури вебзастосунку, схематичні зображення процесу аналізу техніки виконання вправ, таблиці прикладів правильного та неправильного виконання вправ, структурна схема бази даних SQLite для зберігання тренувальних даних, графіки динаміки кількості повторень та навантажень по різних вправах, приклади побудови графіків для аналізу максимальних показників тренувань, візуалізації звітів тренувальної активності користувача.

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	08.04.25-12.04.25	
3	Аналіз літератури з досліджуваної проблеми	13.04.25-17.04.25	
4	Аналіз технічних засобів	18.04.25-27.04.25	
5	Розробка архітектури вебзастосунку	28.04.25-05.05.25	
6	Програмна реалізація	06.05.25-13.05.25	
7	Оформлення пояснювальної записки	14.05.25-20.05.25	
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. Кобилін О. А.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 62 с., 9 табл., 30 рис., 31 джерело.

ВЕБЗАСТОСУНОК, АНАЛІЗ ТРЕНУВАНЬ, FLASK ФРЕЙМВОРК, DOCKER КОНТЕЙНЕРИ, БАЗА ДАНИХ.

Об’єктом роботи є процес збору, зберігання, обробки та аналізу даних про фізичні тренування спортсменів. В рамках роботи було розглянуто особливості організації персонального обліку тренувань, формування бази даних, а також реалізацію аналітичних функцій для подальшого відстеження прогресу та якості виконання вправ. Також проаналізовано особливості структурування інформації для забезпечення швидкого доступу до записаних даних.

Метою роботи є розробка вебзастосунку, за допомогою якого можна зручно вести, зберігати та аналізувати інформацію про тренування користувачів з урахуванням сучасних підходів до веброзробки та контейнеризації програмного забезпечення за допомогою Docker.

У результаті роботи створено універсальний вебзастосунок, який дозволяє ефективно організовувати облік тренувальної активності спортсменів, зберігати історію тренувань, контролювати навантаження, будувати аналітичні графіки, проводити оцінку техніки виконання вправ та створювати звіти для подальшого аналізу прогресу.

WEB APPLICATION, TRAINING ANALYSIS, FLASK FRAMEWORK, DOCKER CONTAINERS, DATABASE.

The object of this work is the process of collecting, storing, processing, and analyzing data on athletes' physical training. In the course of the research, the peculiarities of organizing personal training records, building a database, and implementing analytical functions for further monitoring of progress and exercise technique quality were considered. The peculiarities of data structuring were also analyzed to ensure fast access to the recorded information and the ability to generate statistical reports.

The purpose of this work is to develop a web application that allows users to conveniently record, store, and analyze training information using modern approaches to web development and software containerization with Docker.

As a result of the work, a universal web application was created that efficiently organizes the tracking of athletes' training activities, stores the training history, monitors loads, builds analytical charts, evaluates exercise technique, and generates reports for further progress analysis.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 5
Вступ 7
1 Аналіз предметної області та постановка задачі 8
1.1 Огляд існуючих підходів до аналізу тренувань 8
1.2 Основні технології розробки вебзастосунків 11
1.2.1 Flask фреймворк Python 11
1.2.2 Порівняння баз даних SQLite проти PostgreSQL 12
1.2.3 Docker як інструмент контейнеризації 12
1.2.4 Розпізнавання образів у вебзастосунку 13
1.3 Методи візуалізації даних 15
1.4 Постановка задачі 16
2 Математичне обґрунтування вибраних методів 17
2.1 Формалізація тренувального процесу 17
2.2 Обробка тренувальних даних і фільтрація 19
2.3 Побудова графіків і візуалізації прогресу 20
2.4 Розпізнавання техніки виконання вправ 23
2.5 Комп'ютерний зір у розпізнаванні техніки виконання вправ 27
2.6 Аналіз статистики результатів тренувань 29
3 Реалізація вебзастосунку 33
3.1 Архітектура вебзастосунку 33
3.2 Інтерфейс користувача та маршрутизація 35
3.3 Функціональні можливості вебзастосунку 41
3.4 Структура бази даних вебзастосунку 49
3.5 Контейнеризація та розгортання через Docker 52
3.6 Тестування працездатності та стійкості системи 55
Висновки 58
Перелік джерел посилання 59

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних

Docker – платформа для контейнеризації застосунків

API – Application Programming Interface (інтерфейс взаємодії компонентів)

REST – Representational State Transfer (архітектурний стиль для API)

Flask – Python фреймворк для створення вебзастосунків

SQLite – реляційна база даних

CRUD – Create, Read, Update, Delete (створення, зчитування, оновлення, видалення базові операції з даними)

HTML – HyperText Markup Language (мова розмітки для створення вебсторінок)

CSS – Cascading Style Sheets (таблиці стилів)

CSV – Comma Separated Values (формат табличних даних)

Android – операційна система на мобільних пристроях

iOS – операційна система на мобільних пристроях

UI – User Interface (інтерфейс користувача)

MVC – Model View Controller (архітектурний патерн для структурування вебзастосунків)

SQL – Structured Query Language (мова структурованих запитів до баз даних)

PostgreSQL – серверна система управління базами даних

СУБД – система управління базами даних

MediaPipe – бібліотека для комп’ютерного зору, що дозволяє відстежувати положення частин тіла на зображенні чи відео

OpenCV – бібліотека комп’ютерного зору для роботи із зображеннями, відео та обробки зображень

NumPy – бібліотека для чисельних обчислень

Matplotlib – бібліотека для побудови графіків та візуалізації даних

Pandas – бібліотека для аналізу і обробки табличних даних

Python – мова програмування високого рівня

JSON – JavaScript Object Notation (формат обміну даними у вигляді тексту)

HTTP – HyperText Transfer Protocol (протокол передачі гіпертекстових документів)

RGB – Red Green Blue (модель кодування кольору)

Dockerfile – конфігураційний файл для створення Docker образів

ВСТУП

У сучасному світі фізична активність є невід'ємною частиною здорового способу життя. Кількість людей, що займаються спортом самостійно збільшується – це створює потребу у зручних інструментах для самостійного моніторингу та аналізу результатів своїх тренувань.

Більшість наявних мобільних застосунків для фітнесу пропонують жорстко фіксовані програми або обмежені сценарії використання, що не враховують індивідуальні особливості тренувань користувача такі як кількість підходів, зазначення додаткової ваги, частота тренувань тощо. Крім того подібні застосунки рідко надають деталізовану статистику та можливість візуалізації прогресу.

Розробка власного вебзастосунку дозволяє створити більш універсальний та персоналізований інструмент для збору та аналізу тренувань. Використання сучасних технологій забезпечує зручність у розгортанні на великій кількості різних платформ.

Актуальність роботи полягає у реалізації потреби користувачів у гнучкому та інформативному інструменті для ведення та аналізу тренувань без прив'язки до жорстких фітнес програм. Крім того, розвиток вебтехнологій та контейнеризації, зокрема Docker, відкриває нові можливості для створення кросплатформених застосунків, які легко адаптуються до різних середовищ без необхідності складної установки. Це особливо актуально для індивідуальних користувачів та спортивних клубів, які потребують простого, надійного та масштабованого рішення для відстежування тренувального прогресу. Таким чином, розробка вебзастосунку відповідає сучасним технічним тенденціям та реальним потребам користувачів, поєднуючи простоту використання, функціональність та можливість легко бути кросплатформеним.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Огляд існуючих підходів до аналізу тренувань

Сьогодні ринок мобільних і вебзастосунків, які орієнтовані на фітнес та фізичну активність, стрімко розвивається. Користувачі мать доступ до десятків рішень, які допомагають формувати звички, слідкувати за прогресом та отримувати рекомендації щодо тренувань. Проте, більшість таких застосунків сконцентровані на масовому сегменті користувачів або на вузькому наборі функцій без глибокої можливості персоналізації та аналітики.

Одним із найпоширеніших прикладів є BeStronger – серія програм для Android, яка має такі застосунки як «100 віджимань», «50 підтягувань», «300 присідань» тощо. Ці рішення зосереджені на досягненні конкретної мети за допомогою фіксованої програми. Їх головна перевага це простота у них користувач виконує вправи за вказаним графіком, система намагається поступово підвищити навантаження, але навіть на моєму прикладі використання можна зрозуміти, що ці програми тренувань дуже шаблонні тому іноді навантаження на наступному тренуванні буває зовсім нереалістичним. Такі застосунки не дозволяють гнучко змінювати програму та додавати тренування з додатковою вагою.

Інший приклад Adidas Training by Runtastic, який пропонує широкий вибір вправ з інструкціями, відео та порадами від тренерів. Цей застосунок також включає ркомендації щодо способу життя та харчування. Його сильна сторона це інтерактивність і якісний візуальний контент. Але навіть у такій системі користувач обмежений в кастомізації. Наприклад, він не може зберігати власні схемі тренувань с довільною кількістю підходів, довільною додатковою вагою або коментарями. Статистика виводиться у вигляді простих чисел без можливості глибокого аналізу чи візуалізації динамікі тренувань.

Популярний серед прихильників тренувань без обладнення застосунок Freeletics спеціалізується на тренуваннях з власною вагою. Його алгоритм

формує програму на основі оцінки стану користувача, що позитивно впливає на мотивацію людини, яка користується застосунком. Але, цей застосунок це більше виступає у ролі тренера, ніж збирає та аналізує персональні дані. Користувач не може самостійно додати власну вправу або переглянути графік змін у навантаженні.

Для досвідчених користувачів цікавим варіантом є FitNotes це застосунок орієнтований на ручне введення журналу тренувань. Він дозволяє зберігати тип вправи, кількість підходів, додаткову вагу. Застосунок простий у використанні і не нав’язує власну програму, що є великою перевагою. Проте у застосунку практично повністю відсутня система візуалізації, не реалізовано побудову графіків чи виведення статистики за період. Користувач бачить лише списки записів і не має змоги повноцінно аналізувати свій прогрес.

Nike Training Club є одним з найвідоміших фітнес застосунків, орієнтованих на початківців, так і на більш досвідчених спортсменів. Він надає велику кількість готових тренувань з відеоінструкціями від професійних тренерів. Основна перевага це висока якість контенту, добре структуровані програми та інтеграція з екосистемою Nike. Проте цей застосунок не підтримує ручне введення власних даних про тренування. Також відсутня можливість детального аналізу навантаження чи перегляду прогресу у вигляді графіків. Застосунок виконує переважно роль цифрового тренера, а не інструменту для персонального моніторингу тренувань.

JEFIT це застосунок для силових тренувань, що дозволяє створювати власні програми, відстежувати підходи, повторення та вагу. Він включає базу даних вправ і можливість ведення щоденника тренувань. На відміну від більшості конкурентів цей застосунок надає базову аналітику, таку як об’єм виконаної роботи та графік прогресу. Проте інтерфейс застосунку вважається перевантаженим, а повна функціональність доступна лише у платній версії. Крім того застосунок більше підходить для класичних тренувань у тренажерному залі та не адаптований під заняття з власною вагою або змішані типи тренувань.

Strava є лідером серед застосунків для бігу, їзди на велосипеді та інших видів кардіо тренувань. Застосунок автоматично фіксує маршрут, швидкість, час і дистанцію за допомогою GPS, має розширену систему соціальної взаємодії, наприклад, користувачі можуть ставити лайки, коментувати, змагатися між собою. Проте застосунок в основному фокусується на тренуваннях на відкритому повітрі і не пристосований для ведення силових або функціональних тренувань. Статистика обмежена тільки параметрами руху і не враховує тренування з вагою чи вправи на повторення.

Strong це мінімалістичний застосунок для відстеження силових тренувань. Його головна перевага це інтуїтивно зрозумілий інтерфейс та можливість фіксації підходів, додаткової ваги, повторів та часу відпочинку. Він підтримує створення власних шаблонів тренувань і базову побудову графіків. Незважаючи на це, застосунок не пропонує глибокого аналізу, немає інструментів, щоб подивитися дані конкретний проміжок часу, а візуалізація прогресу реалізована досить обмежено. Крім того, багато функцій доступні лише у платній версії застосунку, що знижує його привабливість для широкого кола користувачів.

Отже, можна зробити висновок, що жоден із розглянутих застосунків не об’єднує всі ключові елементи, які були б корисні для комплексного аналізу індивідуальних тренувань. Частина з них має зручний інтерфейс, інші мають аналітику та кастомізацію, проте жоден не забезпечує одночасно можливості ручного введення даних, підтримку різних типів навантаження, повноцінну візуалізацію прогресу, кросплатформеність. У цьому контексті створення спеціалізованого вебзастосунку, що використовує сучасні технології веброзробки та контейнеризації такі як Flask, Docker, SQlite дозволяє вирішити ці проблеми. Такий застосунок буде орієнтований саме на користувача, який хоче не лише тренуватись, а й бачити реальний прогрес, аналізувати динаміку, порівнювати періоди та приймати рішення на основі даних власних тренувань.

1.2 Основні технології розробки вебзастосунків

1.2.1 Flask фреймворк Python

Flask це мікрофреймворк для мови програмування Python, призначений для розробки вебзастосунків. Його основна особливість полягає в простоті структури та гнучкості, що робить його ідеальним вибором для невеликих або середніх по розміру проєктів. Flask не нав’язує сурової структури у порівнянні з такими як Django. Завдяки цьому розробник має більше контролю над архітектурою застосунку та може адаптувати її під конкретні потреби. Попри свою простоту, Flask підтримує ключові сучасні підходи до розробки, такі як реалізація REST API, підключення баз даних, авторизацію, обробку форм та інші [19, 25].

У застосунках, які використовують Flask часто використовується патерн MVC (Model View Controller). В ньому модель відповідає за обробку даних, вигляд за відображення інформації користувачеві, а контролер за логіку обробки запитів, маршрутизацію та взаємодію між компонентами. Серед переваг Flask можна виділити його здатність до швидкого розгортання на різних платформах, що особливо зручно під час розробки та тестування. Вбудований вебсервер дозволяє розробнику миттєво запускати застосунок без потреби у сторонніх інструментах. Також фреймворк має активну спільноту, яка підтримує велику кількість додаткових бібліотек, зокрема для авторизації, валідації, роботи з базами даних тощо. Ще однією важливою перевагою є сумісність Flask з технологією контейнеризації Docker, що спрощує масштабування та забезпечує стабільне розгортання на різних середовищах без потреби у повторному налаштуванні.

Flask широко використовується в реальних проєктах від REST API до повноцінних вебінтерфейсів. У межах даної роботи він обраний як основа для створення вебзастосунку, що дозволяє зберігати, обробляти та аналізувати дані тренувань спортсменів у зручній та доступній формі [18, 25].

1.2.2 Порівняння баз даних SQLite проти PostgreSQL

Для зберігання інформації про тренування користувачів було обрано систему керування базами даних SQLite. Вона є вбудованою реляційною базою даних, яка не потребує окремого сервера для роботи. Це значно спрощує розгортання проєкту, особливо на ранніх етапах розробки або при використанні у локальному середовищі. Перевагами SQLite є простота налаштування, кросплатформеність та повна автономність. База даних зберігається у вигляді одного файлу, що дозволяє зручно переносити застосунок між середовищами без втрати даних. SQLite підтримує більшість SQL операцій, необхідних для роботи з таблицями та інші [2, 9].

Альтернативним варіантом могла би бути система PostgreSQL це потужна серверна база даних, яка забезпечує масштабованість, розширену підтримку транзакцій, ролей користувачів та складні запити. Проте її використання доцільне переважно у великих проєктах або у випадках коли необхідна висока продуктивність і доступ для великої кількості користувачів.

Отже, було зроблено вибір на користь SQLite, як більш легкої та гнучкої СУБД, яка повністю покриває потреби застосунку, орієнтованого на персональне використання та розгортання у вигляді окремого контейнера.

1.2.3 Docker як інструмент контейнеризації

Docker це платформа з відкритим кодом, що забезпечує контейнеризацію програмного забезпечення. Вона дозволяє запакувати застосунок разом з усіма його залежностями, бібліотеками, конфігураціями та середовищем у спеціальні ізольовані контейнери. Завдяки цьому програма може однаково працювати на будь-якому сервері або комп’ютері незалежно від локальних налаштувань системи. Однією з ключових складових є Dockerfile це конфігураційний файл у якому описано інструкції щодо

створення образу. У ньому вказується базове середовище, встановлюються необхідні бібліотеки, копіюються файли проєкту та задається команда запуску. На основі цього файлу створюється Docker образ, з якого потім запускається контейнер.

Контейнери працюють швидше та легше, ніж традиційні віртуальні машини, бо вони не потребують окремої ОС, вони розділяють ядро з хост-системою, але працюють ізольовано. Це забезпечує надійність, стабільність та передбачуваність поведінки застосунку.

У межах даної роботи Docker використовується для створення ізольованого середовища розгортання вебзастосунку. Завдяки цьому спрощується поширення проєкту, бо користувачу достатньо встановити Docker і запустити контейнер без необхідності окремо встановлювати Flask, SQLite та інші компоненти. Контейнеризація дозволяє масштабувати застосунок у майбутньому або перенести його на сервер без змін у коді.

Отже, використання Docker забезпечує гнучкість, стабільність та універсальність розгортання, що є особливо важливим для практичного застосування розробленого застосунку [19].

1.2.4 Розпізнавання образів у вебзастосунку

Із розвитком комп’ютерного зору та доступністю відповідних бібліотек для Python з’явилась можливість інтеграції технологій розпізнавання образів навіть у невеликі індивідуальні проєкти, у тому числі у фітнес застосунки. Було реалізовано функціонал, що дозволяє проводити базовий аналіз техніки виконання вправ на основі обробки зображень або окремих відеокадрів. Такий підхід відкриває нові можливості для користувача, бо дає змогу оцінювати не лише кількісні характеристики тренувань, як кількість повторень чи кількість додаткової ваги, а й якісні аспекти такі як положення тіла, симетричність, стабільність та дотримання правильної техніки виконання вправ.

Принцип роботи полягає у зчитування вхідного зображення, на якому користувач зображений у процесі виконання вправи та подальшому аналізі положення частин тіла. Для цього використовуються сучасні бібліотеки Python, а конкретно MediaPipe, яка надає модель для визначення ключових точок на тілі людини. Ця модель дозволяє в режимі реального час або з обробкою за кадром визначати положення основних суглобів та сегментів тіла таких як плечі, лікті, таз, коліна, гомілки тощо. Після отримання координат цих точок застосунок виконує геометричні обчислення такі як розрахунок кутів між сегментами або відстаней між точками, які є маркерами правильної чи неправильної техніки.

Система розпізнавання образів може виявити чи користувач достатньо глибоко присідає, чи виконує віджимання з прямою спиною, чи не завалюється корпусом у бік. Для аналізу використовуються заздалегідь визначені межі правильних значень, наприклад, кут у коліні має бути менше ніж 90°, спина повинна бути відносно вертикальна у присіданні тощо. Якщо значення виходить за межі допустимого діапазону, система позначає виконання вправи як некоректне. У випадку задовільної техніки програма підтверджує правильність виконання, що можна відобразити у підсумковій таблиці тренування або у графіку прогресу.

Важливою особливістю обраного підходу є автономність. Усі обчислення виконуються локально, без потреби у зовнішніх сервісах чи підключенні до інтернету, що підвищує приватність даних користувача. Відсутність необхідності в передаванні відео на сторонні сервери є суттєвою перевагою з точки зору безпеки. Технічно, вся обробка здійснюється у середовищі Python із використанням таких бібліотек як OpenCV, NumPy та MediaPipe. OpenCV відповідає за попередню обробку зображення перетворення формату, масштабування, фільтрацію шумів. NumPy використовується для математичних операцій таких як обчислення кутів за координатами точок, аналіз напрямків. MediaPipe основний модуль, що дозволяє визначати координати частин тіла на зображенні [5, 6].

Описаний підхід має високу практичну значущість. Користувач отримує можливість об’єктивно оцінювати техніку своїх рухів, навіть без допомоги тренера. Це підвищує рівень відповідальності, дозволяє уникати хибних патернів руху, які можуть призвести до травм. У результаті застосунок виконує не лише функцію журналу тренувань, але й частково роль віртуального помічника, що дозволяє визначати координати частин тіл на зображенні.

У подальшому можливе розширення цієї функціональності за рахунок навчання моделі на наборі прикладів правильних та неправильних технік з використанням методів машинного навчання. Однак навіть у поточному варіанті інтеграція комп’ютерного зору у систему дозволяє вивести функціональність фітнес застосунку на якісно новий рівень, роблячи акцент не тільки на кількісному, а і на якісному контролі тренувань.

1.3 Методи візуалізації даних

Візуалізація є важливою складовою будь-якої аналітичної системи, оскільки дає змогу наочно відображати динаміку тренувань, зміни у навантаженні, зміну кількості повторень та особисті рекорди користувача. Було використано у вебзастосунку побудову графіків за допомогою засобів Python, зокрема бібліотеки Matplotlib.

Графіки відображають зміну кількості підходів, повторень, додаткової ваги та частоти тренувань у часі. Наприклад, лінійний графік дає змогу побачити прогрес користувача у певній вправі, а стовпчикові діаграми порівняти обсяг навантаження між різними днями чи типами тренувань.

Використання Matplotlib дозволяє зручно налаштовувати вигляд графіків, задавати підписи на вісях координат, підписуючи ключові точки, обирати стиль та кольорову гаму. Це дає змогу адаптувати графічне представлення під індивідуальні потреби користувача [5, 8, 14].

Отже, візуалізація даних підвищує зручність користування, а також сприяє глибшому аналізу тренувального процес, надаючи користувачеві можливості для оцінки ефективності своїх тренувань.

1.4 Постановка задачі

Таким чином, зростаюча популярність самостійних тренувань та недостатня гнучкість наявних фітнес застосунків робить актуальним створення системи, яка дозволяє користувачеві фіксувати, зберігати та аналізувати власну тренувальну активність. Тому ставиться завдання розробки вебзастосунку, що забезпечує введення користувачем даних про тренування, їх збереження у локальній базі даних, візуалізацію прогресу за допомогою графіків.

Об’єктом роботи є процес збору та аналізу даних про тренування спортсменів.

Метою роботи є розробка вебзастосунку, що дозволяє зберігати, обробляти та візуалізувати дані про фізичну активність користувача із застосування технологій Flask, SQLite, Docker.

Для досягнення мети необхідно вирішити такі завдання:

– провести огляд існуючих підходів до розробки фітнес застосунків та методів візуалізації;

– дослідити можливості фреймворку Flask для реалізації вебінтерфейсу;

– обґрунтувати вибір засобів зберігання даних;

– реалізувати систему введення та редагування тренувальних даних;

– реалізувати побудову графіків тренувального прогресу за допомогою засобів Python;

– забезпечити контейнеризацію вебзастосунку з використанням Docker для подальшого зручного розгортання.

2 МАТЕМАТИЧНЕ ОБҐРУНТУВАННЯ ВИБРАНИХ МЕТОДІВ

2.1 Формалізація тренувального процесу

Для побудови ефективної інформаційної системи, що дозволяє аналізувати тренування спортсмена, необхідно формалізувати процес тренування як об’єкт для зберігання та подальшої обробки. Тренування можна

Представити як множину вправ, кожна з яких має ряд параметрів, які впливають на навантаження та результат. Формально одне тренування T складається з n вправ $E_1, E_2, \dots, E_n$, кожна з яких має певну кількість підходів a_i, повторень у кожному підході $r_{i,j}$ та навантаження $\omega_{i,j}$.

Загальне навантаження для вправи i визначається за формулою:

$$V_i = \sum_{j=1}^{a_i} r_{i,j}\, \omega_{i,j}, \tag{2.1}$$

де V_i – навантаження для вправи i;

a_i – кількість підходів у вправі;

$r_{i,j}$ – кількість повторень у підході j;

$\omega_{i,j}$ – навантаження з урахуванням додаткової ваги у підході j.

Загальний об’єм тренувального навантаження визначається як сума по всіх вправах:

$$V_T = \sum_{i=1}^{n} V_i, \tag{2.2}$$

де V_T – загальне навантаження за тренування;

n – кількість вправ у тренуванні;

V_i – навантаження для вправи i.

Середнє навантаження за певний проміжок часу наприклад тиждень, розраховую за формулою:

$$\overline{\mathrm{V}} = \frac{1}{k}\sum_{l=1}^{k} V_{T_l}, \tag{2.3}$$

де $\overline{\mathrm{V}}$ – середнє тренувальне навантаження за період;

k – кількість тренувань у періоді;

V_{T_l} – навантаження під час l-го тренування.

Темп зміни навантаження між послідовними тренуваннями обчислюється за такою формулою:

$$\Delta V_i = V_i^t - V_i^{t-1}, \tag{2.4}$$

де ΔV_i – зміна навантаження у вправі i;

V_i^t – значення навантаження у поточному тренуванні;

V_i^{t-1} – значення навантаження у попередньому тренуванні.

Ці математичні залежності дозволяють кількісно описати тренувальний процес, порівнювати ефективність у різні дні та формалізовано визначати прогрес. На основі цих показників будуються графіки, виводяться особисті рекорди та виявляється спад продуктивності [5, 6, 15].

Таблиця 2.1 – Структура запису тренування

Дата	**Вправа**	**Підходи**	**Повторення**	**Додаткова вага (кг)**
1	**2**	**3**	**4**	**5**
26-05-2025	Віджимання	3	20	0
26-05-2025	Присідання з вагою	4	15	10

Продовження таблиці 2.1

1	2	3	4	5
26-05-2025	Підтягування	3	8	0
26-05-2025	Бруси	2	12	5

2.2 Обробка тренувальних даних і фільтрація

Наступним етапом є реалізація механізмів обробки та фільтрації даних. Ці операції виконуються на стороні серверу із застосуванням SQL запитів до бази даних. Використано найбільш поширені приклади обробки такі як обчислення сумарного навантаження за певний період, середніх значень, частоти тренувань, визначення днів з максимальним об'ємом.

Нехай D – множина дат, що покривають обраний часовий інтервал. Для кожної дати $d_i \in D$ розраховую сумарне навантаження за день:

$$V_{d_i} = \sum_{j=1}^{n_i} V_{ij}, \tag{2.5}$$

де V_{d_i} – сумарне навантаження за день d_i;

n_i – кількість вправ, виконаних у день d_i;

V_{ij} – навантаження за вправу j в день d_i.

Для подальшої обробки даних у системі також реалізовано фільтрацію за типом вправи, наприклад, лише «підтягування» або за категорією – «силові вправи», «кардіо» тощо. Це дозволяє окремо аналізувати навантаження за кожним напрямком тренувального процесу. Також вебінтерфейс передбачає пошук по діапазону дат $[d_{min};\ d_{max}]$, фільрацію по навантаженню, наприклад, лише тренування з додатковою вагою більше 10 кг, визначення найкращих

результатів. Найбільше значення навантаження серед усіх записів тренувань обчисляється як:

$$V_{max} = \max_{i,j} V_{ij}, \tag{2.6}$$

де V_{max} – максимальне навантаження серед усіх вправ;

i – індекс тренування;

j – індекс вправи у тренуванні.

Крім аналітичних операцій важливим є визначення регулярності тренувань. Нехай N – кількість днів в обраному періоді, а $T \subseteq D$ це підмножина дат, коли тренування відбулись. Тоді регулярність можна визначити як відношення кількості активних днів до загальної кількості днів:

$$R = \frac{|T|}{N} * 100\%, \tag{2.7}$$

де R – відсоток регулярності;

$|T|$ – кількість днів з тренуваннями;

N – загальна кількість днів у періоді.

Такі обчислення є базою діаграм активності, таблиць з підсумками за тиждень або місяць, а також виведення особистих рекордів користувача.

Отже, обробка тренувальних даних у межах системи включає як базові аналітичні розрахунки так і фільтрацію по категоріях, датах, ваговому навантаженню. Це забезпечує основу для подальшої візуалізації та аналізу даних.

2.3 Побудова графіків і візуалізації прогресу

Візуалізація є важливою частиною аналітики у вебзастосунку для тренувань, оскільки дозволяє користувачу швидко оцінити динаміку змін,

виявити тенденції у навантаженні так своєчасно скорегувати програму тренувань. Найбільш інформативними для аналізу є графіки темпу прогресу у конкретних вправах. Нехай є впорядкована послідовність тренувань $T_1, T_2, \dots, T_k$, де кожному тренуванню відповідає значення загального навантаження V_{T_i}. Побудова графіка базується на парі значень (x_i, y_i):

$$x_i = d_i, \qquad y_i = V_{T_i}, \tag{2.8}$$

де d_i – дата тренування;

V_{T_i} – загальний об'єм навантаження.

На основі цих значень будую лінійну діаграму, яка показує зміну навантаження в часі. Також додаю графік ковзного середнього, який згладжує короткочасні коливання. Нехай ширина вікна m, тоді значення ковзного середнього визначається як:

$$\bar{V}_{T_i} = \frac{1}{m} \sum_{j=i-m+1}^{i} V_{T_j}, \tag{2.9}$$

де $\bar{V}_{T_i}$ – значення ковзного середнього навантаження для тренування i;

m – кількість попередніх тренувань для обчислення середнього;

V_{T_j} – навантаження для тренування j.

Також можна виводити графік рекордів у певній вправі, наприклад, максимальну кількість повторень або кількість повторень з максимальною додатковою вагою. Нехай R_i – особистий рекорд у вправі i, то графік відображає точки це максимум у межах одного тренування.

$$y_i = \max\left(r_{i,j}\omega_{i,j}\right), \tag{2.10}$$

де y_i – максимальне навантаження за один підхід у вправі i;

$r_{i,j}$ – кількість повторень у підході j;

$\omega_{i,j}$ – додаткове навантаження у підході j.

На рисунку 2.1 наведено приклад графічної візуалізації навантаження за останні 14 днів.

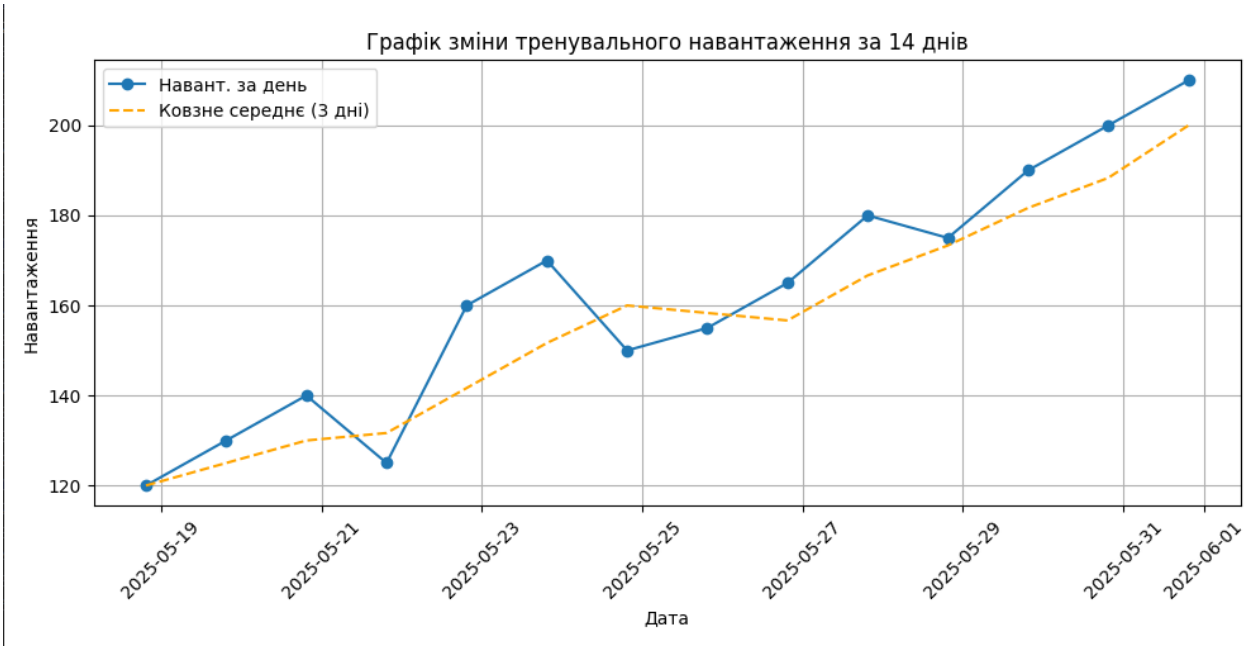

Рисунок 2.1 – Графік зміни тренувального навантаження користувача

Такі графіки будуються за допомогою бібліотеки Matplotlib на стороні Python серверу, а також є можливість формувати їх на клієнтській частині через HTML та SVG. У моєму випадку візуалізація здійснюється на сервері, а зображення у форматі PNG або SVG передається клієнту через вебінтерфейс Flask [21, 22, 31].

Отже, використання графіків дозволяє перетворити числову інформацію у зручну для сприйняття форму, забезпечити зворотний зв’язок та створити дані для аналізу, щоб будувати персональні рекомендації.

2.4 Розпізнавання техніки виконання вправ

Оцінка техніки виконання фізичних вправ є важливою складовою тренувального процесу, що впливає як на ефективність так і на безпеку спортсмена. У рамках розробки вебзастосунку розглядається можливість автоматичного аналізу рухів користувача на основі зображень або відеозаписів з використанням алгоритмів комп'ютерного зору. Для математичного опису рухів тіла людини застосовується концепція ключових точок основних анатомічних орієнтирів таких як суглоби, кінцівки, тулуб. Положення орієнтирів визначаються у просторі зображення. Нехай маємо набір точок:

$$P = \{p_1, p_2, \dots, p_m\}, \tag{2.11}$$

де $p_i = (x_i, y_i)$ – координати i-ї ключової точки на зображенні;

m – загальна кількість ключових точок.

Для визначення техніки виконання вправи використовуються відстані між точками та кути між сегментами. Наприклад, для визначення кута згину ліктя при віджиманні застосовується формула косинуса між векторами:

$$\theta = \arccos\left(\frac{\vec{u} * \vec{v}}{|\vec{u}| * |\vec{v}|}\right), \tag{2.12}$$

де θ – кут між векторами у радіанах;

$\vec{u},\ \vec{v}$ – вектори, що відповідають сегментам руки.

Кожна вправа має специфічний діапазон допустимих кутів. Наприклад, для повного віджимання кут у лікті повинен наближатися до 90 градусів у нижній фазі. Якщо кут виявляється меншим або більшим за припустимий діапазон, техніка вважається некоректною. Результат аналізу представлено у вигляді текстового коментаря, оцінки та графічного відображення неправильної пози. Також є фіксація кількості правильних повторень:

$$N_{valid} = \sum_{i=1}^{n} \delta_i, \tag{2.13}$$

де N_{valid} – кількість правильних повторень;

δ_i – індикатор правильності техніки на i-му кадрі або циклі руху;

n – загальна кількість кадрів.

Таблиця 2.2 – Координати ключових точок і кути згинання у вправі «віджимання»

Кадр	**Плече (x, y)**	**Лікоть (x, y)**	**Зап’ясток (x, y)**	**Кут у лікті**	**Оцінка**
1	(240, 140)	(240, 180)	(260, 220)	90,0	правильна
2	(222, 138)	(242, 178)	(263, 215)	87,3	правильна
3	(225, 135)	(250, 170)	(270, 205)	80,1	гранична
4	(230, 130)	(255, 165)	(280, 200)	74,5	погана
5	(220, 145)	(240, 185)	(260, 230)	95,6	правильна
6	(218, 148)	(238, 188)	(258, 235)	98,2	правильна
7	(216, 150)	(236, 190)	(256, 240)	100,5	погана
8	(222, 140)	(242, 180)	(262, 220)	90,2	правильна
9	(224, 138)	(244, 178)	(264, 215)	86,5	правильна
10	(228, 135)	(248, 175)	(270, 210)	82,1	гранична
11	(232, 132)	(252, 170)	(275, 205)	78,0	гранична
12	(235, 130)	(255, 168)	(280, 200)	74,0	погана
13	(238, 128)	(258, 165)	(282, 198)	70,8	погана
14	(220, 142)	(240, 182)	(260, 225)	92,3	правильна
15	(218, 144)	(238, 184)	(258, 228)	94,7	правильна
16	(216, 147)	(236, 187)	(256, 232)	96,9	гранична
17	(214, 149)	(234, 189)	(254, 236)	99,4	погана

Оцінка техніки ґрунтується на діапазоні між 80 та 95 градусами, що відповідає правильному положенню рук у нижній фазі віджимання. Динаміка кута у лікті під час віджимань (рис. 2.2).

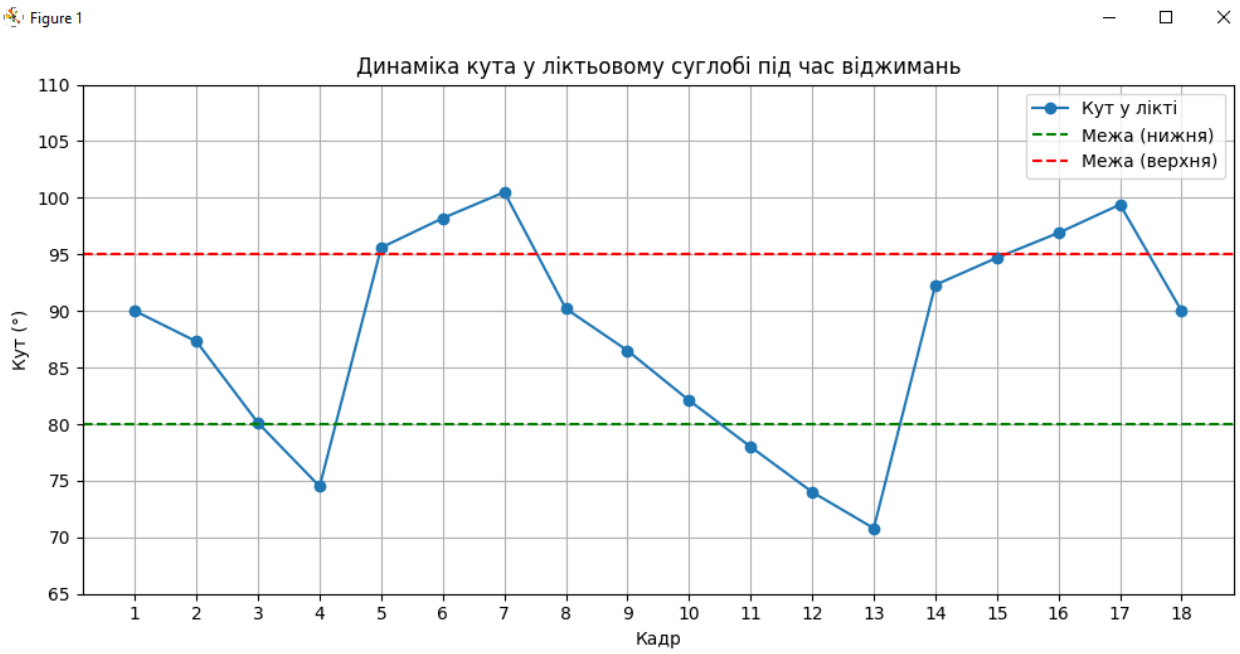

Рисунок 2.2 – Динаміка кута у ліктьовому суглобі під час віджимань

Таблиця 2.3 – Координати ключових точок і кути згинання у вправі «присідання»

Кадр	**Таз (x,y)**	**Коліно (x, y)**	**П’ята (x, y)**	**Кут у коліні**	**Оцінка**
1	**2**	**3**	**4**	**5**	**6**
1	(250, 220)	(250, 270)	(250, 310)	90,0	правильна
2	(248, 222)	(248, 272)	(248, 312)	92,5	правильна
3	(245, 225)	(245, 275)	(245, 315)	95,0	правильна
4	(240, 230)	(242, 278)	(244, 318)	98,6	гранична
5	(238, 235)	(240, 280)	(242, 320)	102,4	погана
6	(236, 238)	(238, 282)	(240, 322)	105,1	погана
7	(234, 240)	(236, 285)	(238, 324)	108,0	погана

Продовження таблиці 2.3

1	2	3	4	5	6
8	(240, 230)	(242, 278)	(244, 318)	98,1	гранична
9	(245, 225)	(245, 275)	(242, 320)	95,2	правильна
10	(250, 220)	(250, 220)	(240, 322)	90,0	правильна
11	(248, 222)	(248, 272)	(238, 324)	91,7	правильна
12	(246, 224)	(246, 274)	(246, 314)	93,3	правильна
13	(243, 227)	(243, 277)	(243, 317)	96,4	гранична
14	(240, 230)	(241, 279)	(242, 319)	99,2	гранична
15	(238, 233)	(239, 281)	(240, 321)	103,7	погана
16	(235, 236)	(237, 283)	(238, 323)	105,5	погана
17	(232, 238)	(235, 286)	(236, 326)	108,8	погана
18	(240, 230)	(242, 278)	(244, 318)	98,0	гранична

Оцінка техніки ґрунтується на діапазоні між 85 та 100 градусами, де кут в коліні вважається оптимальним для повноцінного присідання. Динаміка кута у коліні за кадрами (рис 2.3).

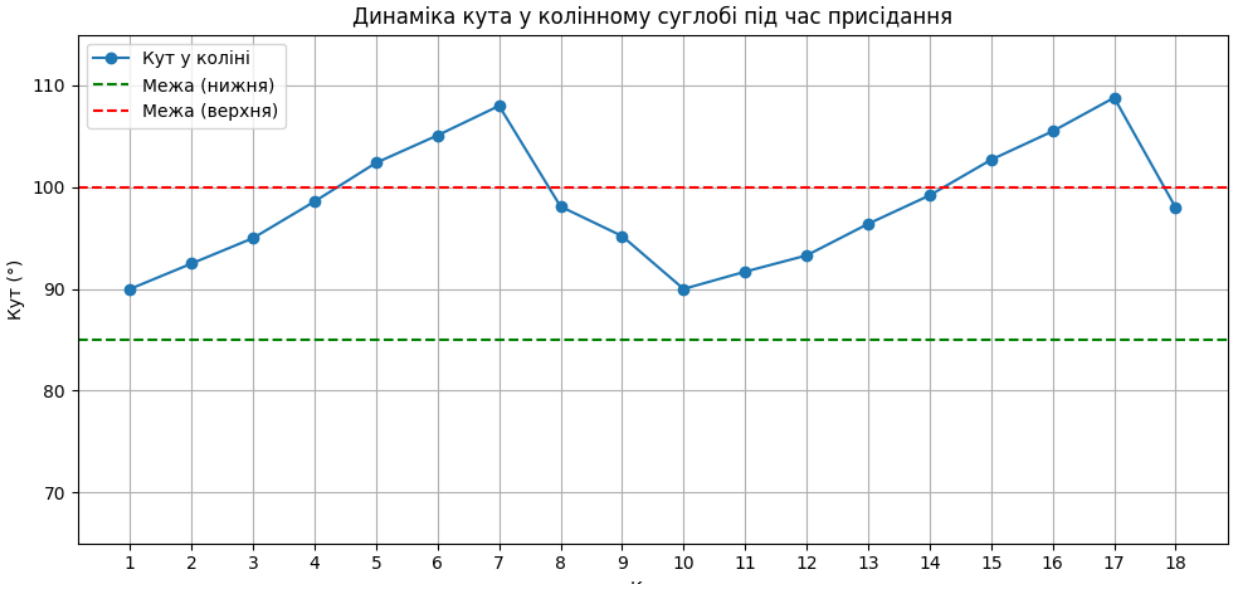

Рисунок 2.3 – Динаміка кута у колінному суглобі під час присідання

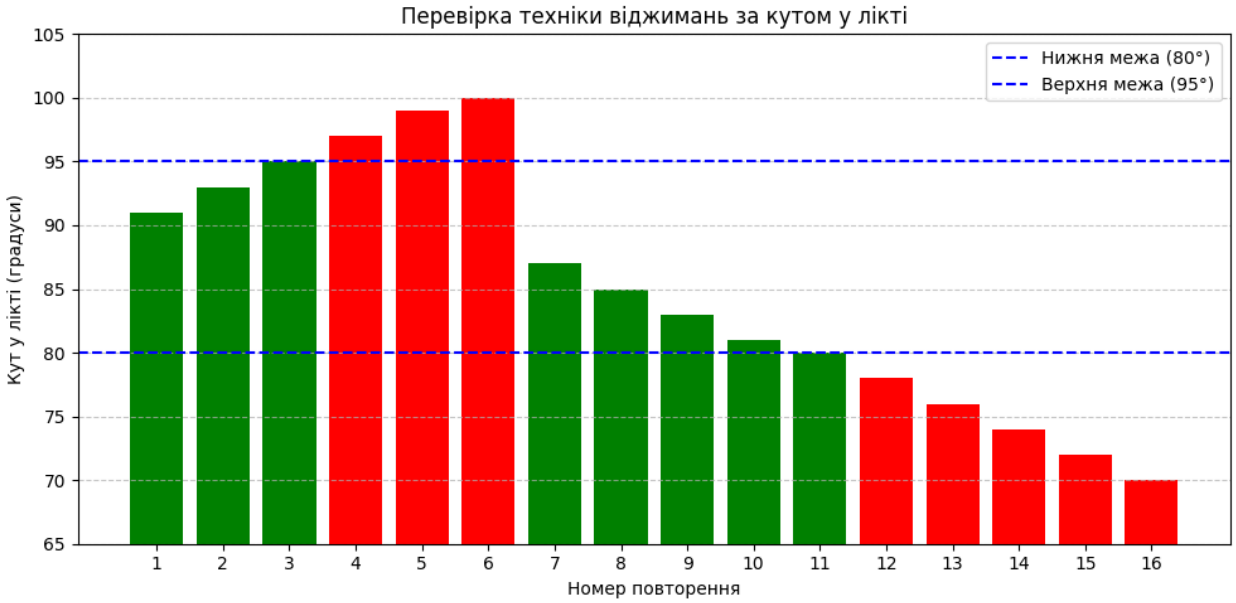

Рисунок 2.4 – Перевірка техніки віджимань за кутом у лікті

2.5 Комп'ютерний зір у розпізнаванні техніки виконання вправ

Комп'ютерний зір це галузь штучного інтелекту, що займається автоматичною інтерпретацією візуальної інформації з цифрових зображень або відео. У моєму випадку з тренуваннями це дозволяє виявляти позиції тіла спортсмена, обчислювати кути кути у суглобах, визначати фазу виконання вправи та перевіряти правильність техніки. Для обробки було використано цифрові зображення це прямокутна матриця пікселів кожен з яких має координати (x, y) та значення яскравості та кольору. Для кольорового зображення з трьома каналами RGB кожен піксель описується трійкою:

$$I(x, y) = \left(R_{xy}, G_{xy}, B_{xy}\right), \tag{2.14}$$

де $I(x, y)$ – значення кольору пікселя, розташованого на координатах (x, y);

$R_{xy} \in [0, 255]$ – інтенсивність червоного каналу;

$G_{xy} \in [0, 255]$ – інтенсивність зеленого каналу;

$B_{xy} \in [0, 255]$ – інтенсивність синього каналу.

В обробці зображень використовується перетворення в градації сірого:

$$I_{gray}(x, y) = 0{,}299R_{xy} + 0{,}587G_{xy} + 0{,}114B_{xy}, \quad (2.15)$$

де $I_{gray}(x, y)$ – значення яскравості у відтінках сірого для пікселя з координатами (x, y).

Це перетворення дозволяє зменшити обсяг обчислень це реально особливо корисно при обробці у реальному часі або на пристроях з обмеженими ресурсами [2, 3, 7, 14].

Основним етапом аналізу тіла людини є знаходження ключових точок, що відповідають суглобам і частинам тіла. Типова модель містить 17–33 ключових точок такі як голова, шия, плечі, лікті, таз, коліна тощо. Найбільш поширеними бібліотеками є MediaPipe та OpenPose перша легка і точно модель від Google, працює у реальному часі навіть на процесорі, друга потужна бібліотека, яка будує скелет людини з максимальною точністю. Нехай модель виявляє m ключових точок, тоді їх координати можна описати як:

$$P = \{p_1, p_2, \dots, p_m\}, \quad (2.16)$$

$$p_i = (x_i, y_i), \quad (2.17)$$

де p_i – координата i-ї точки;

x_i, y_i – піксельні координати на зображенні.

Після отримання координат ключових точок будується сегментна модель це умовний скелет, де відрізки з’єднують пари точок такі як плече та лікоть, лікоть та зап’ясток тощо. За цими сегментами обчислюються кути:

$$\theta = \arccos\left(\frac{\vec{u} * \vec{v}}{|\vec{u}| * |\vec{v}|}\right), \quad (2.18)$$

де θ – кут у суглобі;

$\vec{u}, \vec{v}$ – вектори між трьома суміжними точками.

Кожна вправа має допустимі діапазони для відповідних кутів. При аналізі руху алгоритм перевіряє, чи входить обчислений кут у заданий діапазон. Якщо входить, то техніка вважається правильною. Крім того розраховується час фази між опусканням та підйомом і тривалість повтору:

$$t_{rep} = t_{end} - t_{start} \quad (2.19)$$

де t_{rep} – час фази між опусканням та підйомом;

t_{end} – часова позначка завершення фази;

t_{start} – часова позначка початку фази.

2.6 Аналіз статистики результатів тренувань

Для аналізу правильності техніки виконання вправ застосовується алгоритм, що порівнює виміряний кут у суглобі з допустимим значенням у діапазоні. Для вправи «віджимання» за еталон прийнято інтервал від 80° до 95° у межах якого техніка вважається правильною. Окрім перевірки техніки виконання вправ, вебзастосунок також надає користувачу розширену статистику, що стосується інтенсивності та навантаження під час тренувань. Такий аналіз допомагає оцінити динаміку фізичного розвитку, виявити перенавантаження, побудувати індивідуальні програми тренувань. У таблицях нижче наведено приклади вхідних даних для аналізу віджимань та присідань. Кожен рядок описує один підхід, який включає кількість повторів та використану додаткову вагу [11, 25, 26].

Таблиця 2.4 – Результати віджимань з навантаженням

№ Підходу	Кількість повторів	Додаткова вага (кг)
1	50	0
2	45	8
3	40	20
4	30	20
5	60	0
6	35	12
7	42	15
8	38	10
9	44	5
10	50	0
11	33	18
12	37	16
13	39	10
14	41	12
15	55	0

Таблиця 2.5 – Результати присідань з навантаженням

№ Підходу	Кількість повторів	Додаткова вага (кг)
1	**2**	**3**
1	20	0
2	25	0
3	30	10
4	35	15
5	40	15
6	32	10
7	28	12

Продовження таблиці 2.5

1	2	3
8	30	15
9	22	0
10	24	0
11	26	10
12	29	12
13	33	15
14	38	15
15	36	15

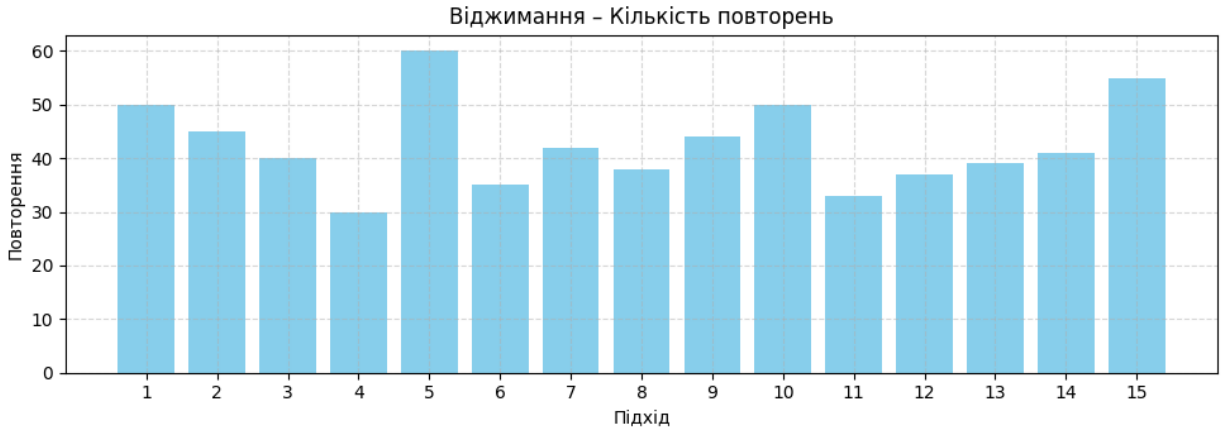

Рисунок 2.5 – Діаграма кількості повторень віджимань

Рисунок 2.6 – Діаграма кількості додаткової ваги у віджиманнях

Рисунок 2.7 – Діаграма умовного навантаження у віджиманнях

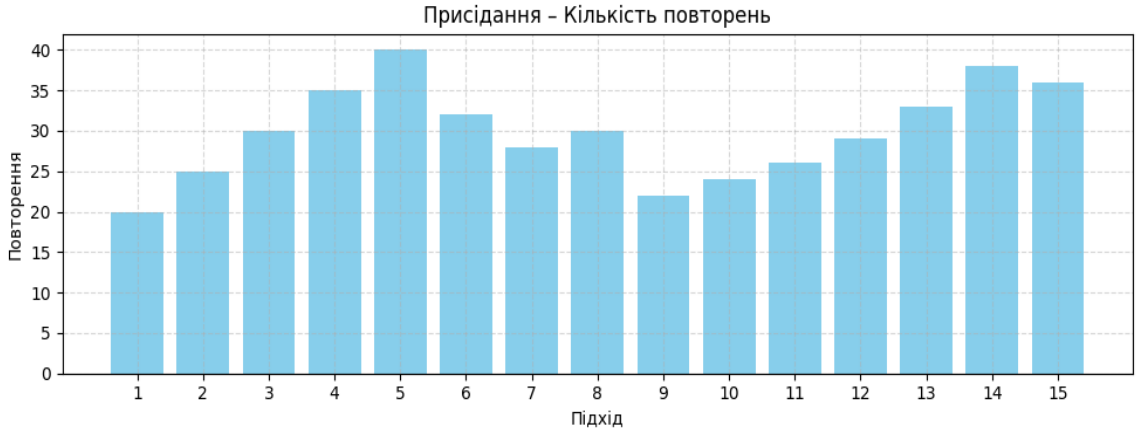

Рисунок 2.8 – Діаграма кількості повторень присідань

Рисунок 2.9 – Діаграма кількості додатковою ваги у присіданнях

3 РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ

3.1 Архітектура вебзастосунку

Архітектура вебзастосунку побудована за класичною схемою клієнт-сервер, яка є однією з найбільш популярних для вебпроєктів завдяки чіткій структурі та легкості розширення. Вебзастосунок складається з таких компонентів як клієнтська частина, вона відповідає за взаємодію користувача з системою через браузер. Включає інтерфейс користувача, створений на основі HTML, CSS, JavaScript та шаблонів Jinja2. Серверна частина побудована на базі фреймворку Flask на мові програмування Python, вона забезпечує обробку HTTP запитів, роботу з даними та логіку застосунку. База даних реалізовано через SQLite як легку, ефективну та портативну систему зберігання даних для тренувань та результатів аналізу. Docker контейнеризація забезпечує легкість та гнучкість розгортання застосунку на різних платформах, що є надзвичайно важливим для забезпечення зручності подальшого використання [19, 25].

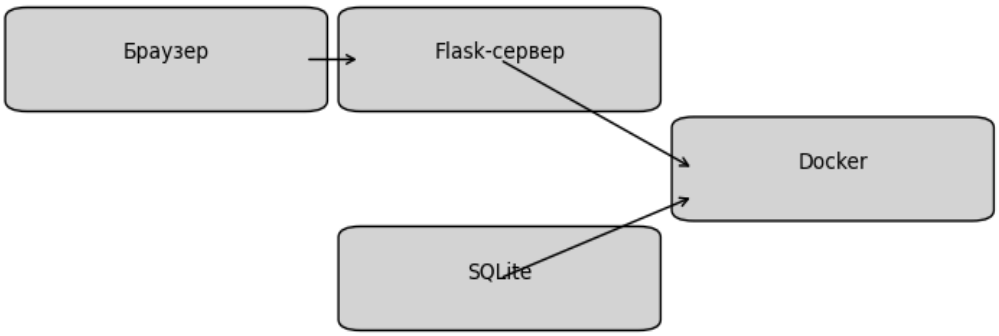

Рисунок 3.1 – Архітектурна схема вебзастосунку

Клієнтську частину побудував за допомогою HTML шаблонів Jinja2, що дозволяє динмічно формувати вебсторінки на основі даних отриманих з сервера. Для стилізації використано CSS, що забезпечує сучасний вигляд та адаптивність венбінтерфейсу для використання на будь-яких пристроях, від комп’ютерів до смартфонів. Для інтерактивності застосував код на JavaScript, який відповідає за надсилання асинхронних запитів на сервер, що забезпечує плавну взаємодію між користувачем та вебзастосунком без перезавантаження сторінки.

Серверна частина побудована на основі фреймворку Flask, що є легким, швидким та гнучким рішенням. Flask забезпечує прийом HTTP запитів їх обробку та відправку результатів назад користувачеві. Реалізовано такі функції серверної частини як отримання запитів від користувача, валідація вхідних даних, обробка та збереження у базі даних SQLite, виконання розрахунків та формування статистики, генерування графіків та зображено з використанням Python бібліотек Matplotlib та Pandas [8, 14].

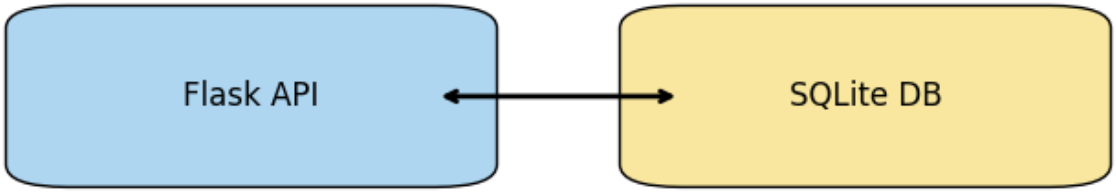

Рисунок 3.2 – Взаємодія Flask API з базою даних SQLite

База даних SQLite використовується як компактна, високошвидкісна і портативна СУБД. Головною перевагою SQLite є відсутність окремого сервера, оскільки вона працює у вигляді окремого файлу, що значно полегшує перенесення між середовищами розгортання.

Docker є технологією, яка дозволяє упакувати всі частини вебзастосунку такі як сервер, базу даних та необхідні залежності в один або за потреби в декілька контейнерів. Контейнеризація вирішує проблему «це не працює на моєму комп’ютері, телефоні тощо», забезпечуючи абсолютно однакове середовище виконання на будь-якій платформі.

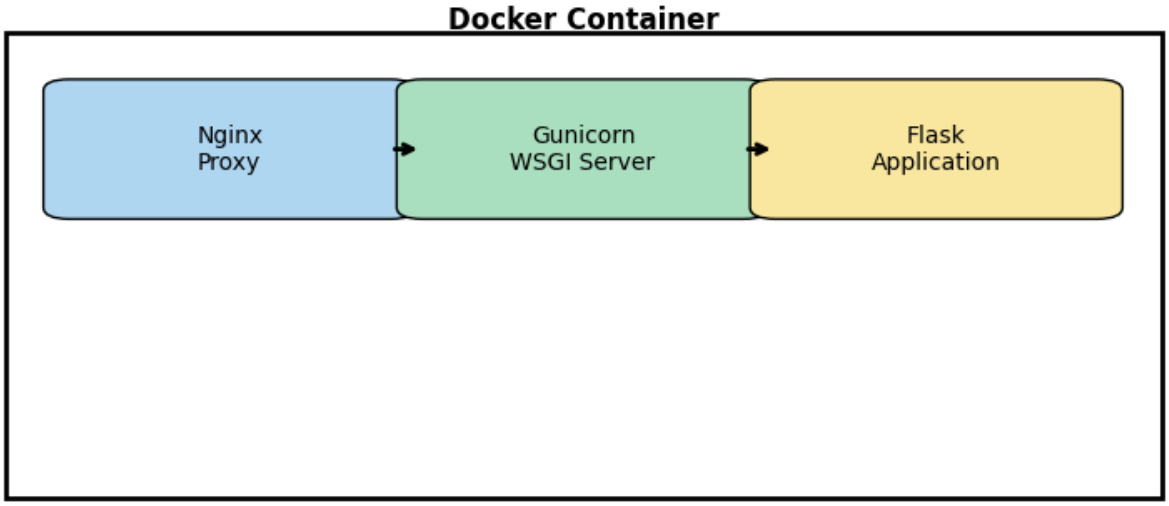

Рисунок 3.3 – Схема розгортання Docker контейнеру

3.2 Інтерфейс користувача та маршрутизація

Одним із важливих аспектів реалізації вебзастосунку є створення зручного, інтуїтивно зрозумілого та функціонального інтефейсу користувача. Для досягнення цієї мети було розроблено вебінтерфейс, який складається з кількох ключових сторінок, що покривають всі основні функції застосунку такі як додавання тренувань, перегляд статистики тренувань, перегляд загальної інформації та персональних налаштувань. Вебзастосунок містить кілька ключових сторінок для різних завдань.

На головній сторінці розташовані елементи навігації, такі як меню з посиланням на інші сторінки застосунку, коротка інформація про останні тренування, а також швидкий доступ до додавання нового тренування.

Також є сторінка додавання тренування вона має зручну форму для введення даних тренування, де користувач вводить дату, тип тренування, вибирає вправи зі списку та зазначає кількість підходів, повторень та додаткову вагу. Інтерфейс розроблено так щоб мінімізувати ймовірність помилок під час введення даних за допомогою валідації на стороні клієнта.

На сторінці статистики користувач може переглядати детальну інформацію про виконані тренування у вигляді таблиць та графіків. Реалізовано можливість перегляду статистики за різні періоди, а також можливість вибору конкретних вправ для аналізу.

Таблиця 3.1 – Дані про підтягування

Підхід	**Кількість повторень**	**Амплітуда руху (%)**	**Контроль тіла (%)**	**Час**
1	10	97	99	1,44
2	11	93	95	1,57
3	9	96	92	1,24
4	11	88	94	1,28
5	11	97	97	1,22
6	8	90	98	1,33
7	9	94	93	1,36
8	9	91	91	1,31
9	9	96	94	1,53
10	9	90	92	1,34
11	11	92	98	1,31
12	10	90	94	1,42
13	11	94	92	1,26

Рисунок 3.4 – Приклад нижньої фази підтягувань від автора застосунку

Рисунок 3.5 – Приклад верхньої фази підтягувань від автора застосунка

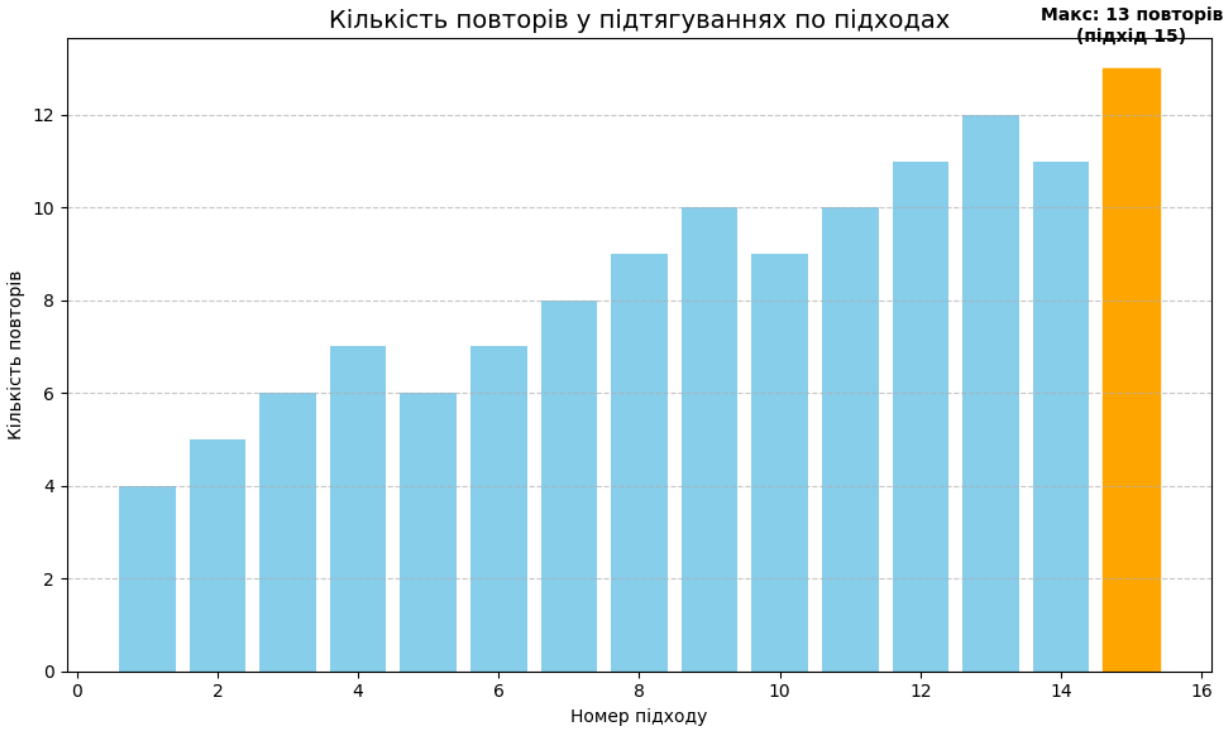

Рисунок 3.6 – Діаграма кількості повторів у підтягуваннях з виділенням максимальної кількості повторів

Вебзастосунок було реаалізовано за допомогую фреймворку Flask, який забезпечує гнучку маршрутизацію шляхом зв’язування URL адрес із конкретними функціями, які відповідають за обробку запитів і повернення відповідей. Маршрутизація вебзастосунку має головну сторінку «/», додавання нового тренування «/add_workout», перегляд статистики «/stats», персональні налаштування «/settings». Для кращого розуміння логіки маршрутизації наведу приклад Python коду, що ілюструє основні шляхи маршрутизації:

Лістинг 3.1 – Основні шляхи маршрутизації у коді:

```
from flask import Flask, render_template, request

app = Flask(__name__)

@app.route('/')
```

```
def home():
    return render_template('home.html')

@app.route('/add_workout', methods=['GET', 'POST'])
def add_workout():
    if request.method == 'POST':
        #логіка додавання до бази даних
        return render_template('success.html')
    return render_template('add_workout.html')

@app.route('/stats')
def stats():
    #отримання даних з бази та передача у шаблон
    data = {'workouts': []}
    return render_template('stats.html', data=data)

@app.route('/settings', methods=['GET', 'POST'])
def settings():
    if request.method == 'POST':
        #збереження змін налаштувань
        pass
    return render_template('settings.html')

if __name__ == '__main__':
    app.run(debug=True)
```

Застосунок також має надання REST API, що дозволяє програмно взаємодіяти з вебзастосунком через HTTP запити. JSON формат відповіді від сервера має наступний вигляд [25, 27]:

Лістинг 3.2 – Приклад відповіді від сервера у форматі JSON:

```
{
  "user_id": 78,
  "workouts": [
    {
      "date": "2025-06-01",
      "exercises": [
        {
          "type": "віджимання",
          "repetitions": 50,
          "weight": 10
        },
        {
          "type": "присідання",
          "repetitions": 60,
          "weight": 20
        }
      ]
    }
  ]
}
```

3.3 Функціональні можливості вебзастосунку

Розроблений вебзастосунок для моніторінгу та аналізу фізичних тренувань реалізує широкий спектр функціональних можливостей, які забезпечують зручну роботу з даними користувача та надають інструменти для глибокого аналізу процесу. Завдяки використанню Python, Flask, SQLite та інших сучасних технологій, застосунок є простим у використанні та водночас

підтримує складну логіку обробки тренувальної інформації. Однією із головних можливостей системи є створення записів про тренування. Користувач має змогу після кожного тренування внести детальну інформацію про виконані вправи такі як назва вправи, кількість повторень, кількість підходів, наявність або відсутність додаткової ваги, а також коментарі щодо самопочуття або техніки виконання. Це дозволяє сформувати повноцінний щоденник тренувань, який у майбутньому можна використовувати, щоб аналізувати та коректувати план тренувань. Також у програмі є приклади виконання найбільш поширених вправ.

Рисунок 3.7 – Перша фаза виконання віджимань від автора програми

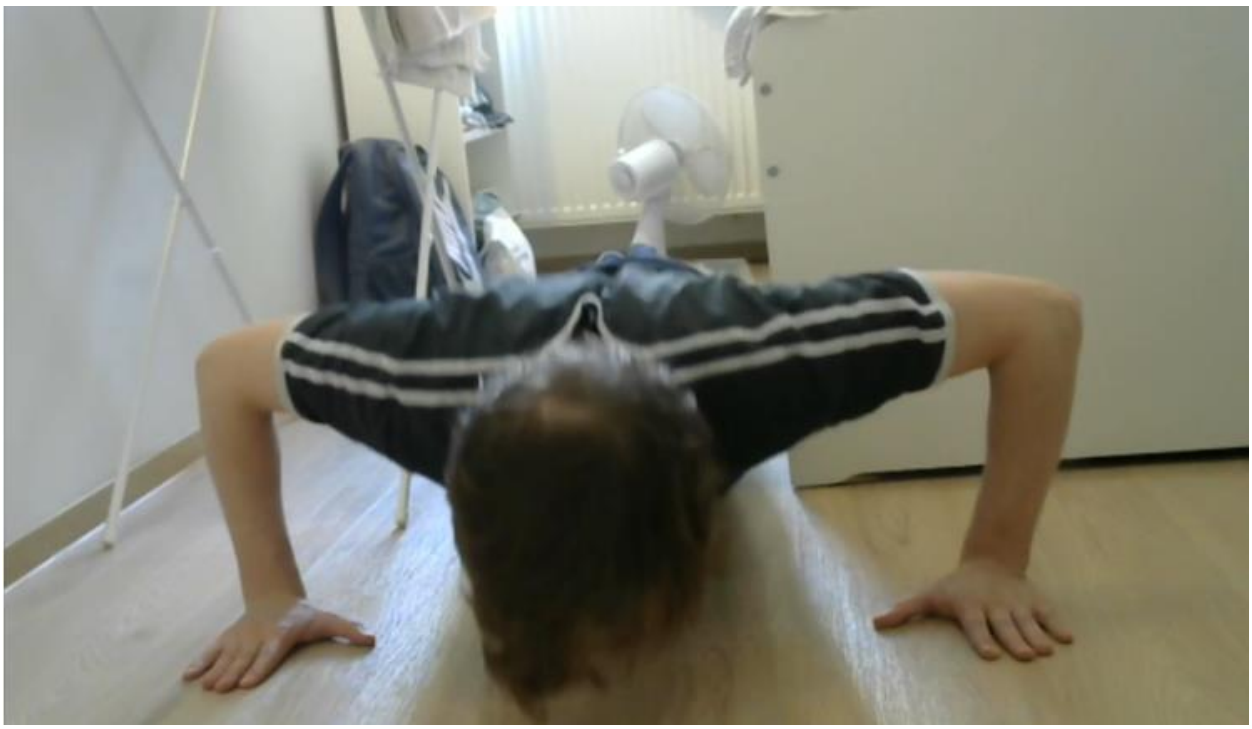

Рисунок 3.8 – Друга фаза виконання віджимань від автора програми

Рисунок 3.9 – Перша фаза виконання присідань від автора програми

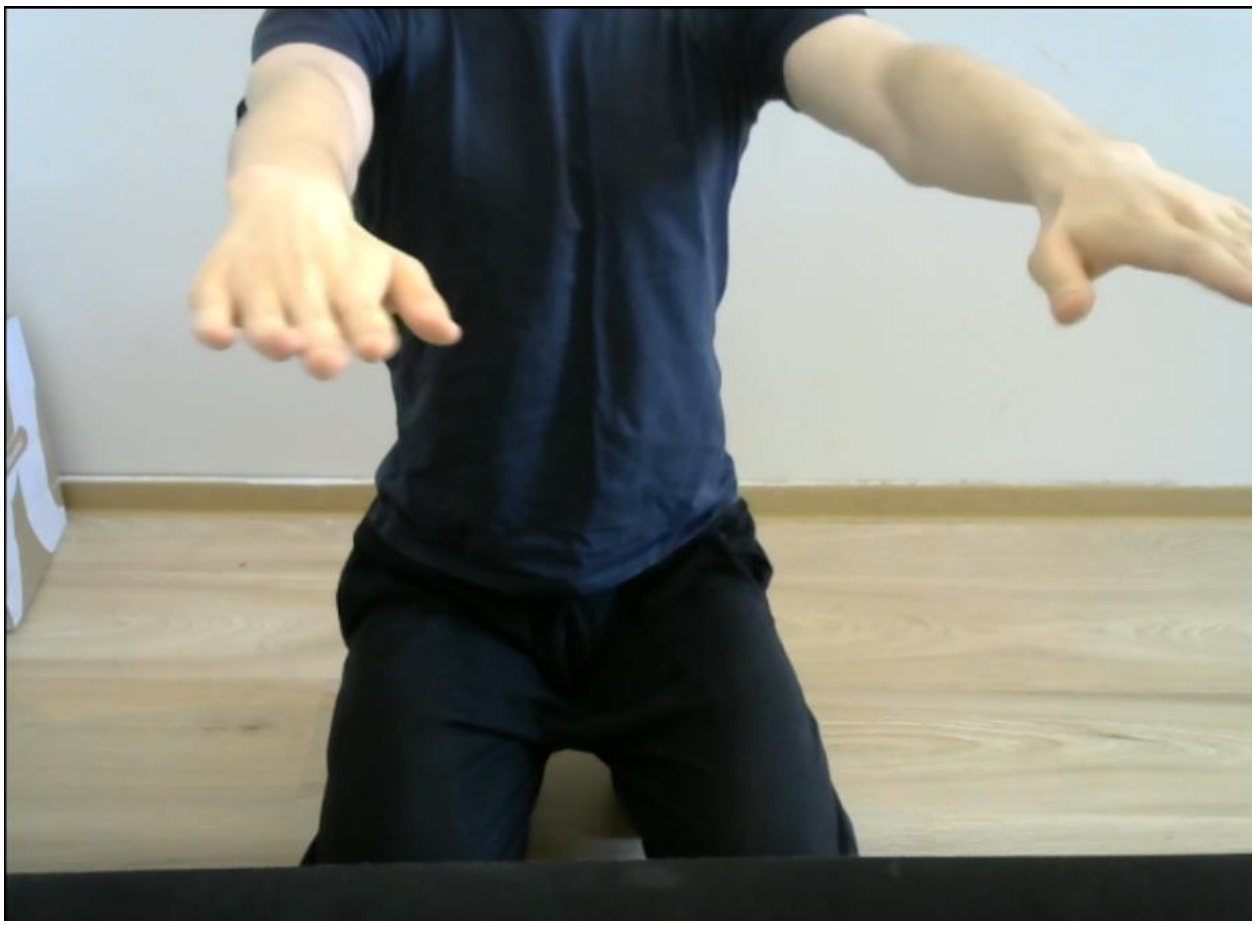

Рисунок 3.10 – Друга фаза виконання присідань від автора програми

Наступною важливою функцією є редагування записів. У випадку помилкового вводу або зміни плану тренування користувач може оновити інформацію, а також видалити непотрібні записи. Ця гнучкість дозволяє підтримувати базу даних у актуальному стані та уникати плутанини. У застосунку є функція розрахунку навантаження. Під навантаженням мається на увазі умовний показник, що визначається добутком кількості повторів на додаткову вагу. Наприклад, виконання 10 присідань з додатковою вагою 20 кг дає додаткове навантаження в 200 умовних одиниць. Якщо вага відсутня, у розрахунок береться лише кількість повторів з своєю власною вагою. Це дуже цікавий параметр, бо він дозволяє зрозуміти не тільки частоту занять, а й їхню інтенсивність.

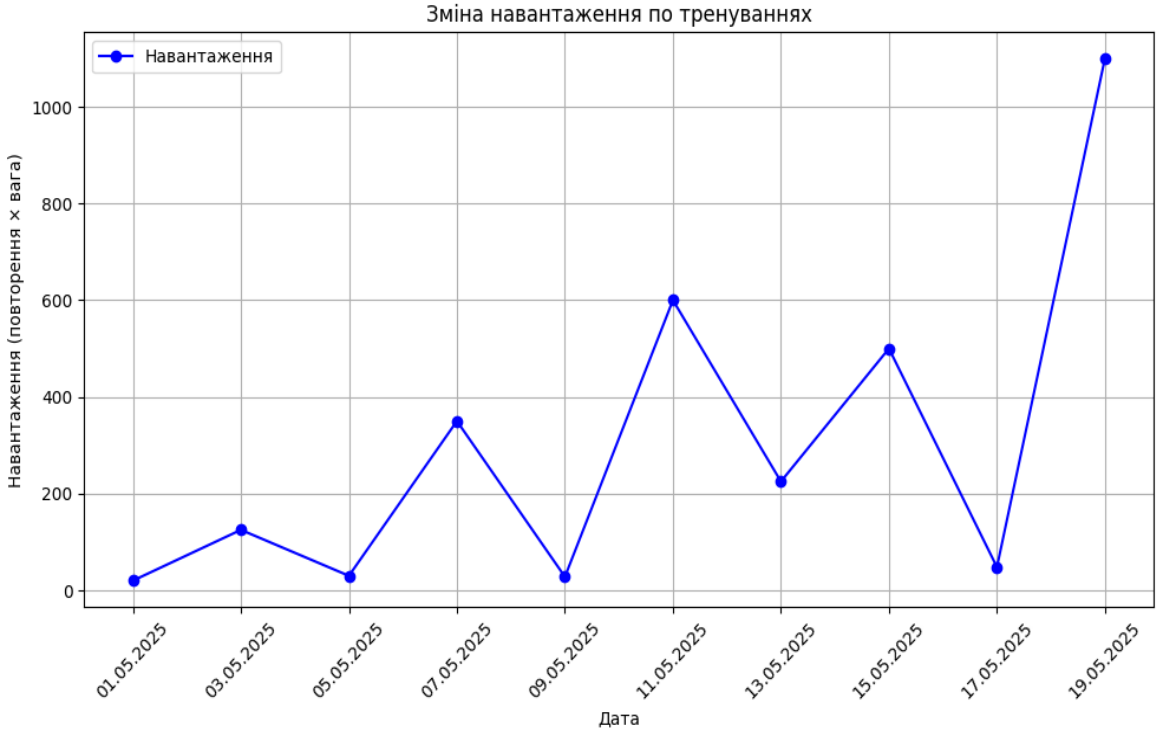

Рисунок 3.11 – Графік зміни навантаження у тренуваннях

Окремої уваги заслуговує функція візуалізації прогресу. Застосунок будує графіки за кількома параметрами такі як кількість повторень, середнє навантаження на тренування, а також виділяє рекордні показники. Можна побачити в який день було виконано максимальну кількість повторень або коли було зафіксовано найбільше навантаження. Це мотивує користувача до подальшого вдосконалення та дозволяє вчасно помітити ознаки перевтоми чи застою. Наприклад, користувач може вибрати для аналізу окремий місяць, тиждень або навіть конкретний діапазон дат. Це дає змогу більш гнучко відстежувати динаміку розвитку фізичних показників, а також оцінити ефективність внесених змін у програму тренувань. Додатково, графіки інтерактивні, тому при наведенні відображаються точні числові значення, що робить аналіз зручним навіть для користувачів без спеціальних знань у сфері спорту чи статистики. Також враховано фактор коректного відображення даних при можливих пропусках тренувань або нерівномірному завантаженні, що забезпечує об’єктивність представленої інформації.

Рисунок 3.12 – Максимальне навантаження за останні 10 тренувань

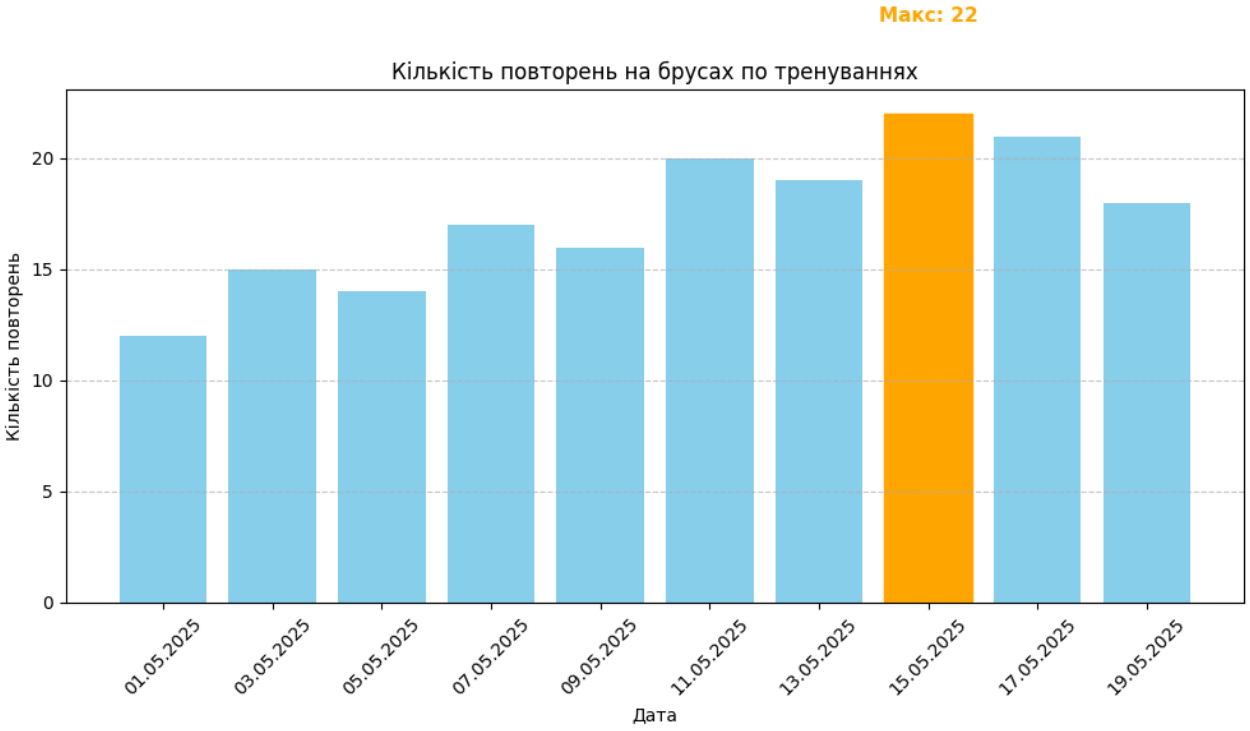

Рисунок 3.13 – Кількість повторень під час тренування на брусах

Окремо потрібно відмітити функцію аналізу техніки виконання вправ. Застосунок може зберігати координати тіла спортсмена на фотографіях під час

виконання вправи. За допомогою цих координат оцінюється правильність техніки. Наприклад, під час присідань аналізується кут у колінному суглобі, положення спини та глубина присідання. Якщо кут присідання не доходить до 90°, система показує підказку. Ця функція вдосконалена за допомогою бібліотек машинного навчання та комп'ютерного зору, таких як OpenCV та MediaPipe [3, 24, 27].

Рисунок 3.14 – Схематичне зображення позиції тіла при віджиманні

Таблиця 3.2 – Приклади правильного та неправильного виконання віджимань

№	Ознака виконання	Правильне виконання	Неправильне виконання	Коментар
1	**2**	**3**	**4**	**5**
1	Положення голови	Голова на одній лінії з тілом	Голова опущена вниз або задерта вгору	Неправильне положення викликає навантаження на шию
2	Спина	Пряма, без прогину	Прогнута або зігнута	Прогин у спині може спричинити травму попереку

Продовження таблиці 3.2

1	2	3	4	5
3	Руки	Розташовані трохи ширше плечей	Занадто широко або вузько	Невірне положення рук зменшує ефективність або збільшує ризик травми
4	Лікті	Знаходяться під кутом 45°	Розходяться у сторони під 90°	Велике навантаження на плечові суглоби
5	Таз	В одній площині з тілом	Провисає або задертий догори	Знижує ефективність вправи
6	Коліна	Прямі, не згинаються	Зігнуті	Людина зменшує амплітуду і змінює техніку
7	Опора на долоні	Повна, рівномірна	Опора лише на пальці або на основу долоні	Зменшується стабільність положення
8	Дихання	Вдих вниз, видих вверх	Нерівномірне або затримка дихання	Погіршує продуктивність та може стати причиною запаморочення
9	Швидкість виконання	Контрольований рух, приблизно 2-3 секунди на цикл	Занадто швидко або дуже повільно	Швидкі ривки це втрата контролю, а занадто повільні призводять до перевантаження

Продовження таблиці 3.2

1	2	3	4	5
10	Повна амплітуда	Груди майже торкаються підлоги	Недостатній нахил вниз	Недостатня амплітуда зменшує навантаження на м’язи

Функція експорту даних у форматі CSV. Користувач у будь-який момент може зберегти свої записи у вигляді таблиці, що дозволяє виконувати подальшу обробку у Microsoft Excel або Google Sheets. Це особливо зручно для тренерів або фахівців, які працюють з декількома клієнтами.

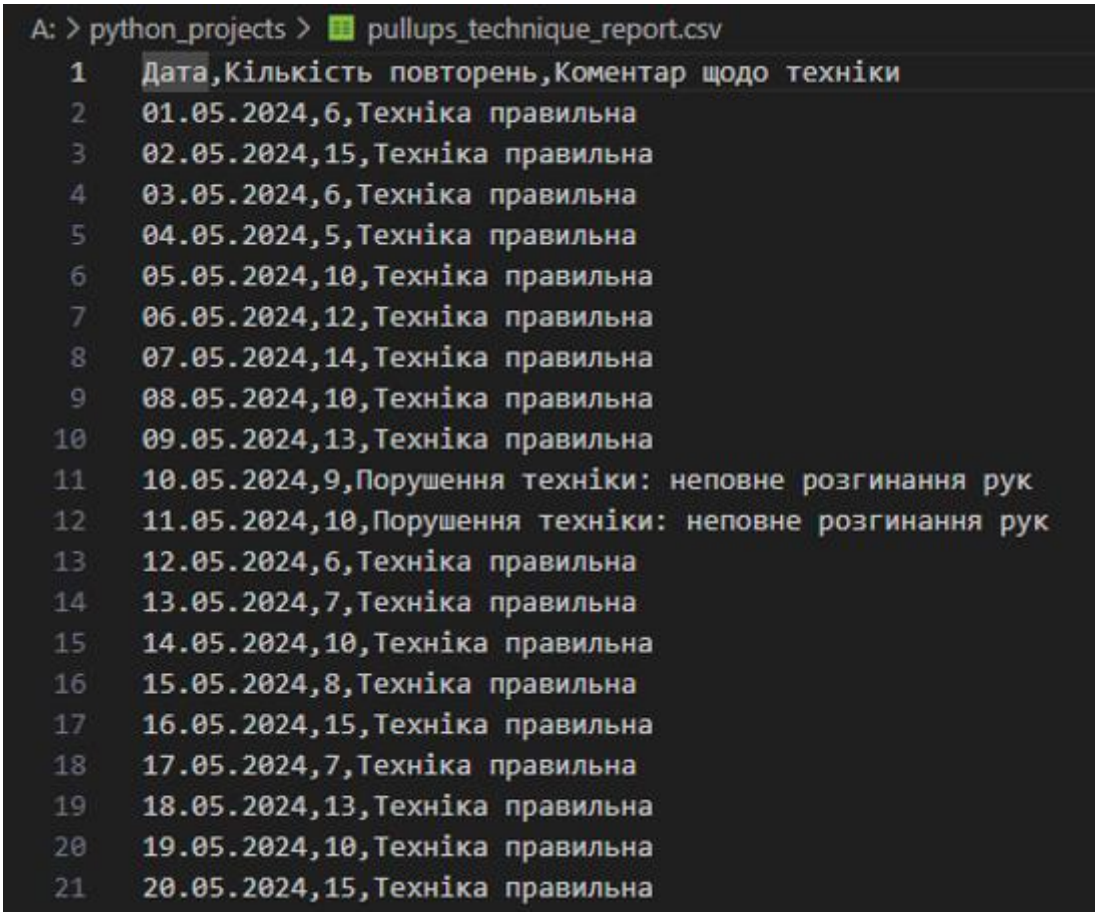

Рисунок 3.15 – Приклад CSV таблиці у тренуваннях підтягувань

3.4 Структура бази даних вебзастосунку

Вебзастосунок використовує реляційну базу даних SQLite, яка дозволяє ефективно зберігати, оновлювати й аналізувати інформацію про тренування спортсменів. Структура бази даних створена так, щоб забезпечити гнучкість при роботі з різними типами тренувань та зберігати деталізовані дані для подальшого аналізу. Основною одиницею зберігання інформації є таблиці, які розподілені за функціональним призначенням.

Таблиця «Users» має такі поля:

– user_id – унікальний ідентифікатор користувача;
– username – ім’я користувача;
– email – адреса електронної пошти користувача;
– password_hash – зашифрований пароль;
– registration_date – дата реєстрації.

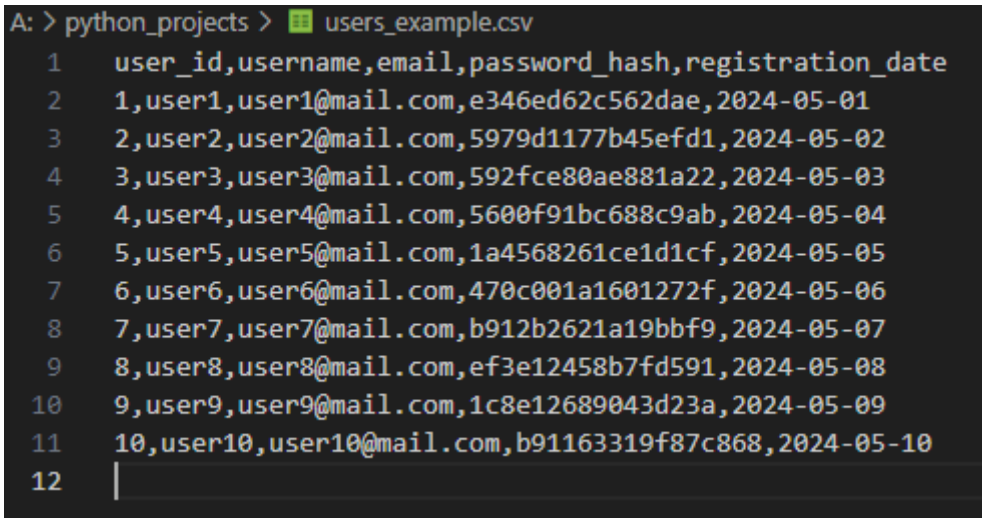

Рисунок 3.16 – Приклад CSV таблиці

Таблиця «Exercises» має такі поля:

– exercise_id – унікальний ідентифікатор тренування;
– name – назва тренування;
– description – опис вправи.

```
exercise_id,name,category,description
1,Підтягування,Силова,Опис вправи підтягування.
2,Віджимання,Силова,Опис вправи віджимання.
3,Присідання,Силова,Опис вправи присідання.
4,Планка,Силова,Опис вправи планка.
5,Біг,Силова,Опис вправи біг.
6,Випади,Силова,Опис вправи випади.
7,Бруси,Силова,Опис вправи бруси.
8,Біцепс,Силова,Опис вправи біцепс.
9,Тяга,Силова,Опис вправи тяга.
10,Жим,Силова,Опис вправи жим.
```

Рисунок 3.17 – Приклад CSV таблиці

Таблиця «Workout_Details» має такі поля:

– detail_id – унікальний ідентифікатор оцінки;
– workout_id – ідентифікатор тренування;
– exercise_id – ідентифікатор вправи;
– weight – додаткова вага, якщо використовується;
– technique_note – коментар щодо техніки виконання вправи.

A: > python_projects > workout_details_example.csv

```
detail_id,workout_id,exercise_id,sets,repetitions,weight,technique_note
1,4,6,3,14,10,Гарна амплітуда
2,10,2,1,13,10,Техніка відмінна
3,12,6,5,7,5,Техніка відмінна
4,11,3,4,10,10,Лікті занадто широко
5,14,10,3,14,10,Техніка відмінна
6,13,7,1,6,15,Лікті занадто широко
7,13,7,1,7,20,Покращити контроль
8,14,2,2,9,5,Лікті занадто широко
9,9,8,5,14,5,Покращити контроль
10,5,2,5,20,15,Стабільно
11,10,6,5,15,5,Гарна амплітуда
12,14,3,2,10,20,Стабільно
13,2,9,1,17,15,Техніка відмінна
14,10,8,2,18,0,Техніка відмінна
15,6,4,5,15,5,Лікті занадто широко
16,9,8,1,8,0,Стабільно
17,6,3,3,19,10,Стабільно
18,9,10,3,11,10,Стабільно
19,5,1,4,13,10,Покращити контроль
20,10,2,3,5,15,Лікті занадто широко
```

Рисунок 3.18 – Приклад CSV таблиці

Таблиця «Technique_Asessment» має такі поля:

– assessment_id – унікальний ідентифікатор оцінки;
– detail_id – ідентифікатор запису у деталях тренувань;
– image_path – шлях до файлу із зображенням виконання вправи;
– correctness_score – оцінка техніки виконання від 0 до 100;
– comments – коментарі згенеровані алгоритмом.

A: > python_projects > technique_assessment_example.csv

```
assessment_id,detail_id,image_path,correctness_score,comments
1,13,media/tech1.jpg,93,Техніка ідеальна
2,5,media/tech2.jpg,86,Трохи порушена форма
3,11,media/tech3.jpg,95,Недостатній розгін рук
4,20,media/tech4.jpg,89,Супер!
5,16,media/tech5.jpg,91,Техніка ідеальна
6,4,media/tech6.jpg,87,Техніка ідеальна
7,19,media/tech7.jpg,98,Потрібен контроль спини
8,13,media/tech8.jpg,85,Супер!
9,13,media/tech9.jpg,94,Супер!
10,3,media/tech10.jpg,90,Техніка ідеальна
11,11,media/tech11.jpg,85,Потрібен контроль спини
12,10,media/tech12.jpg,78,Недостатній розгін рук
13,16,media/tech13.jpg,84,Недостатній розгін рук
14,1,media/tech14.jpg,78,Техніка ідеальна
15,9,media/tech15.jpg,100,Недостатній розгін рук
```

Рисунок 3.19 – Приклад CSV таблиці

3.5 Контейнеризація та розгортання через Docker

Контейнеризація це сучасний підхід, який дає змогу об’єднувати програмний код, бібліотеки та залежності в єдине ізольоване середовище, яке називається контейнером. Docker є провідною технологією в цій галузі, забезпечуючи ефективне розгортання застосунків на будь-якій платформі. Завдяки Docker застосунок стає портативним, легким у масштабуванні та захищеним від конфліктів залежностей, що суттєво спрощує роботу

розробників та адміністраторів систем. Docker ґрунтується на трьох основних поняттях такі як образи, контейнери, та Docker файли. Образ Docker це шаблон, який використовується для створення контейнерів. Контейнер є конкретним екземпляром образу. Dockerfile це спеціальний текстовий файл, що містить інструкції для створення образу. Dockerfile містить команди які вказують, як саме потрібно зібрати Docker образ.

Лістинг 3.3 – Dockerfile для вебзастосунку на Flask:

```
FROM python:3.10-slim

WORKDIR /app

COPY requirements.txt ./
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

EXPOSE 5000

CMD ["python", "app.py"]
```

У цьому файлі:

– FROM – задає базовий образ, який містить середовище Python;
– WORKDIR – встановлює робочу директорію всередині контейнера;
– COPY – транспортує файли з локальної машини у контейнер;
– RUN – виконує команди для встановлення залежностей;
– EXPOSE – відкриває порт для доступу до вебзастосунку;
– CMD – запускає програму, коли контейнер стартує.

Для спрощення управління кількома контейнерами одночасно використовується інструмент Docker Compose, який дозволяє описати всі

контейнери та їх налаштування в одному YAML файлі. Docker Compose особливо корисний, коли застосунок має додаткові сервіси, такі як бази даних або кеші.

Лістинг 3.4 – файл docker-compose.yml:

```
version: '3.8'
services:
  web:
    build: .
    ports:
      - "5000:5000"
    volumes:
      - .:/app
    depends_on:
      - db

  db:
    image: sqlite
    volumes:
      - db-data:/var/lib/sqlite

volumes:
  db-data:
```

Цей конфігураційний файл описує два сервіси такі як вебзастосунок на Flask та базу даних SQLite. За допомогою команди docker-compose up запускаються обидва сервіси автоматично.

Контейнери Docker ізолюють застосунок від основної системи, що забезпечує стабільність роботи незалежно від середовища запуску. Легко можна створювати та зупиняти додаткові контейнери, які допомагають

швидко масштабувати застосунок. Docker гарантує що застосунок однаково буде працювати на різних платформах і серверах. Контейнери запускаються за кілька секунд, що дозволяє швидко реагувати на зміни та оновлення.

Отже, контейнеризація через Docker суттєво спрощує життєвий цикл вебзастосунку, роблячи процес розгортання та підтримки набагато ефективнішим. Вона дозволяє розробникам зосередитися на написанні коду, а адміністраторам на керуванні сервісами та інфраструктурою. Docker є важливим інструментом для будь-якого сучасного вебпроєкту, який потребує високої продуктивності, стабільності та простоти масштабування.

3.6 Тестування працездатності та стійкості системи

У межах тестування було перевірено такі речі, як додавання нових записів про тренування, відображення графіків, збереження у базі даних, стійкість системи при швидкому введенні даних, відображення помилок у випадку введення некоректної інформації, правильність роботи розпізнавання правильної техніки віджимань. Тестування проводилось вручну з використанням базових скриптів на Python, що робили запити до API та взаємодію з інтерфейсом.

Таблиця 3.3 – Результати тестування

№	**Дія**	**Очікуваний результат**	**Фактичний результат**	**Статус**
1	**2**	**3**	**4**	**5**
1	Введення тренування з правильними даними	Запис зберігається у базу даних	Запис збережено	Пройдено

Продовження таблиці 3.3

1	2	3	4	5
2	Введення тренування з пустими полями	Помилка введення	Помилка була показана	Пройдено
3	Відображення графіку тренувань віджимань	Графік побудовано	Графік відображено	Пройдено
4	Надсилання 10 записів за 1 секунду	Всі записи оброблені	Успішно	Пройдено
5	Введення тексту в числове поле	Помилка введення	Повідомлення про помилку	Пройдено
6	Неправильна техніка віджимань	Коментар від програми про техніки виконання	Повідомлення про неправильну техніку виконання	Пройдено

У рамках тестування працездатності вебзастосунку було проведено функціональну перевірку основних сценаріїв роботи системи. Таблиця містить перелік ключових дій, які виконувалися під час тестування, а також очікувані та фактичні результати. Усі функціональні модулі були протестовані на коректність обробки даних.

Для визначення меж працездатності було виконано серію запитів до сервера з частотою 5, 10, 20 та 50 запитів на секунду. Метою було встановити, як система себе проводить при активному використанні та чи не призводить це до втрати даних чи падіння застосунку.

Таблиця 3.4 – Результати тесту навантажень

Інтенсивність (зап./сек)	**Середній час відповіді**	**Відхилення**	**Помилки**
5	123 мс	3 мс	0
10	135 мс	6 мс	0
20	158 мс	10 мс	0
50	250 мс	25 мс	1 (тайм-аут)

Результати показали, що вебзастосунок стабільно функціонує при звичайному навантаженні, а також здатен обробляти інтенсивну взаємодію без втрати працездатності. Було встановлено, що застосунок починає втрачати стабільність при кількості запитів понад 50 в секунду, що є прийнятним для персонального або невеликого кола користувачів.

Особливо ретельно перевірено функцію розпізнавання правильного виконання техніки вправ, бо саме ця функціональність є інноваційною складовою роботи застосунку. Під час тестів спеціально імітувалися ситуації, коли користувач виконує віджимання з різними відхиленнями від правильної техніки виконання. Алгоритм стабільно розпізнає й інформує про помилки, такі як недостатня амплітуда руху, неправильна постановка рук чи прогин у спині.

Також було проведено тестування взаємодії користувачів з інтерфейсом програми. Було проведено аналіз простоти використання, коректності повідомлень про помилки, а також зрозумілості навігації між різними розділами застосунку.

Отже, комплексне тестування продемонструвало готовність вебзастосунку до реальних умов експлуатації, що підтверджує відповідність розробленого продукту поставленим технічним завданням і очікуванням користувачів.

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і реалізований вебзастосунок для збору, збереження та аналізу даних про тренування спортсменів з використанням сучасних технологій, таких як Python, Flask, SQLite та Docker. Основна ідея полягала у створенні універсального рішення, яке дозволяє не лише фіксувати основні параметри тренувань, а й здійснювати візуальний аналіз прогресу, будувати графікі, здійснювати оцінку техніки виконання вправ та формувати звіти.

Протягом розробки було проаналізовано існуючі підходи до побудови фітнес застосунків, виявлено їх переваги та недоліки, на основі чого сформульовано основні вимоги до системи. Було обґрунтовано вибір інструментів і методів від використання Flask, як гнучкого фреймворку до впровадження автоматичної побудови графіків за допомогою бібліотеки Matplotlib. Особливістю реалізованого рішення є модуль розпізнавання техніки виконання вправ, який базується на аналізі координатних даних, що допомагає використовувати методи комп’ютерного зору.

Було реалізовано зручний вебінтерфейс, який дозволяє користувачу вручну вводити дані про тренування, додавати кількість повторів, вагу, а також переглядати результати у вигляді наочних графіків. Система формує звіти, які можна отримати у вигляді CSV файлів або збереженні як зображення.

Тестування системи підтвердило її працездатність і стабільність під час типового та інтенсивного використання. Результати показали, що застосунок здатен обробляти запити без втрати даних, а інтеграція Docker дозволила легко його розгортати на різних середовищах.

У процесі була досягнута поставлена мета – створення зручного, функціонального засобу для аналізу тренувань спортсменів. Впроваджено розпізнавання рухів за відео для розпізнавання техніки виконання вправ.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Kobylin, O., Vyskrebentseva, S., & Petrova, R. (2019). Обробка даних, що містять пропуски в задачах кластеризації. *Системи управління, навігації та зв'язку. Збірник наукових праць*, *5*(57).

2. Mashtalir, V., Ruban, I., & Levashenko, V. (Eds.). (2019). Advances in Spatio-Temporal Segmentation of Visual Data (Vol. 876). Springer Nature.

3. Kobylin, O. A., Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Peredrii, O. O. (2020). The application of non-parametric statistics methods in image classifiers based on structural description components. Telecommunications and Radio Engineering, 79(10).

4. Gorokhovatskyi, V., & Tvoroshenko, I. (2023). Identification of visual objects by the search request. *International scientific symposium «INTELLIGENT SOLUTIONS-S»* (pp. 25-27).

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

6. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., and 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.

7. Ibrahim, D. Y., Gorokhovatskyi, V., Tvoroshenko, I., & Mujahed, A. D. (2022). Classification of Images Based on a System of Hierarchical Features.

8. Daradkeh, Y.I., Gorokhovatskyi, V., Tvoroshenko, I., Gadetska, S., and Al-Dhaifallah, M. (2021) Methods of Classification of Images on the Basis of the Values of Statistical Distributions for the Composition of Structural Description Components, *IEEE Access*, 9, pp. 92964-92973.

9. Rabotiahov, A., Kobylin, O., Dudar, Z., & Lyashenko, V. (2018, February). Bionic image segmentation of cytology samples method. In *2018 14th*

International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET) (pp. 665-670). IEEE.

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

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

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

13. 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

27. Gorokhovatskyi, V., Chmutov, Y., Tvoroshenko, I., & Kobylin, O. (2025). Reducing computational costs by compressing the structural description in image

classification methods. Advanced Information Systems, 9(1), 5–12. https://doi.org/10.20998/2522-9052.2025.1.01

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

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

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

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