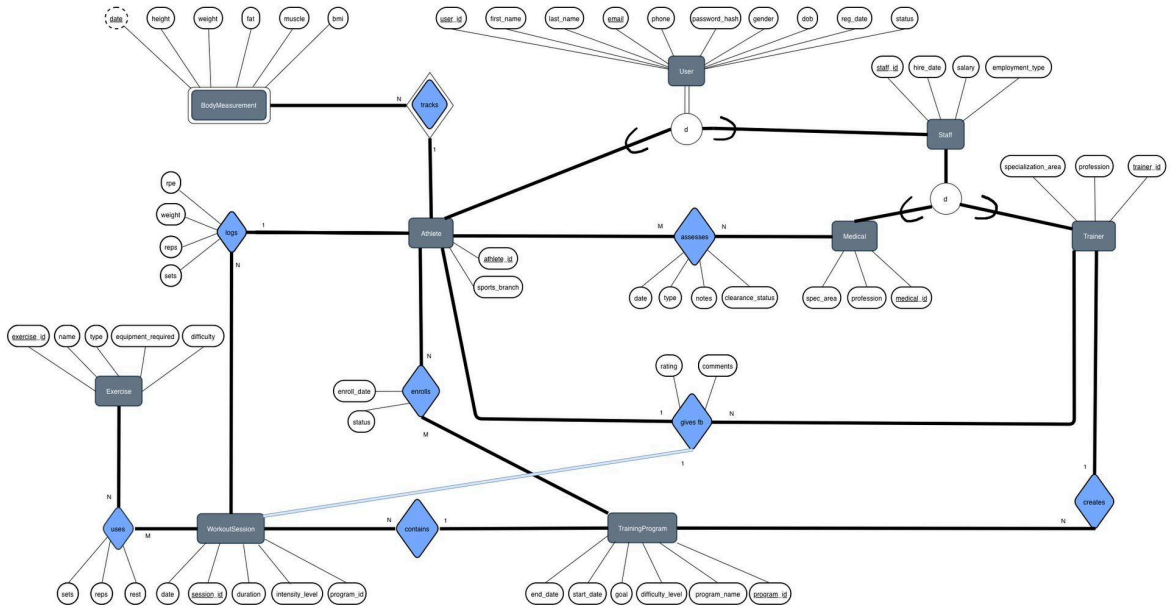
Training Performance and Medical Status Tracking App for Professional Sports Clubs (S-PORT)

Abdullah Nawaz, Efe Yılmaz, Eren Karakuş, Hasan Karagülle, İrfan Can Beşer, Maximilian Kunkel

Project Description

This project focuses on the development of a web-based Gym Management System designed to support the operational needs of a fitness center by integrating athletes, trainers, and medical staff within a unified platform. The system allows athletes to enroll in training programs, follow structured workout sessions, and track physical measurements over time. Trainers can create and manage training programs, schedule workout sessions, monitor athlete progress, and provide feedback to improve training outcomes. In addition, medical staff such as doctors, physiotherapists, and dietitians can record medical assessments, evaluate athlete readiness, and maintain health-related clearance information to ensure safe participation in physical activities. All system data is stored in a MySQL relational database and accessed through a Flask-based web API, enabling consistent data management, role-based access, and realistic modeling of gym workflows.

Entity-Relationship Diagram



Relational Database Design

```
DROP DATABASE IF EXISTS gym;
CREATE DATABASE IF NOT EXISTS gym
CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;USE gym;
```

```sql
CREATE TABLE `User` (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    email VARCHAR(255) NOT NULL UNIQUE,
    phone VARCHAR(20),
    password_hash VARCHAR(255) NOT NULL,
    gender ENUM('male', 'female', 'other'),
    date_of_birth DATE,
    registration_date DATETIME,
    status ENUM('active', 'inactive') DEFAULT 'active'
) ENGINE=InnoDB;

CREATE TABLE Athlete (
    athlete_id INT PRIMARY KEY,
    sports_branch VARCHAR(100),
    CONSTRAINT fk_athlete_user
        FOREIGN KEY (athlete_id) REFERENCES `User`(user_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
) ENGINE=InnoDB;

CREATE TABLE Staff (
    staff_id INT PRIMARY KEY,
    hire_date DATE,
    salary DECIMAL(10,2),
    employment_type ENUM('full-time', 'part-time') NOT NULL,
    CONSTRAINT fk_staff_user
        FOREIGN KEY (staff_id) REFERENCES `User`(user_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
) ENGINE=InnoDB;

CREATE TABLE Trainer (
    trainer_id INT PRIMARY KEY,
    specialization VARCHAR(100),
    experience_years INT,
    CONSTRAINT fk_trainer_staff
        FOREIGN KEY (trainer_id) REFERENCES Staff(staff_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
) ENGINE=InnoDB;

CREATE TABLE Medical (
    medical_id INT PRIMARY KEY,
    profession ENUM('doctor', 'physiotherapist', 'dietitian') NOT NULL,
    specialization_area VARCHAR(100),
    CONSTRAINT fk_medical_staff
        FOREIGN KEY (medical_id) REFERENCES Staff(staff_id)
```

```
        ON DELETE CASCADE
        ON UPDATE CASCADE
) ENGINE=InnoDB;

CREATE TABLE TrainingProgram (
    program_id INT AUTO_INCREMENT PRIMARY KEY,
    program_name VARCHAR(150) NOT NULL,
    difficulty_level ENUM('beginner', 'intermediate', 'advanced'),
    goal TEXT,
    start_date DATE,
    end_date DATE,
    created_by_trainer INT,
    CONSTRAINT fk_program_trainer
        FOREIGN KEY (created_by_trainer) REFERENCES Trainer(trainer_id)
        ON DELETE SET NULL
        ON UPDATE CASCADE
) ENGINE=InnoDB;

CREATE TABLE ProgramEnrollment (
    athlete_id INT,
    program_id INT,
    enrollment_date DATE DEFAULT (CURRENT_DATE),
         completion_status  ENUM('ongoing',  'completed',  'dropped')  DEFAULT
'ongoing',
    PRIMARY KEY (athlete_id, program_id),
    CONSTRAINT fk_enrollment_athlete
        FOREIGN KEY (athlete_id) REFERENCES Athlete(athlete_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT fk_enrollment_program
        FOREIGN KEY (program_id) REFERENCES TrainingProgram(program_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
) ENGINE=InnoDB;

CREATE TABLE WorkoutSession (
    session_id INT AUTO_INCREMENT PRIMARY KEY,
    program_id INT NOT NULL,
    session_date DATE NOT NULL,
    duration INT,
    intensity_level ENUM('low', 'medium', 'high'),
    CONSTRAINT fk_session_program
        FOREIGN KEY (program_id) REFERENCES TrainingProgram(program_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
) ENGINE=InnoDB;

CREATE TABLE Exercise (
    exercise_id INT AUTO_INCREMENT PRIMARY KEY,
```

```
    exercise_name VARCHAR(150) NOT NULL,
    type VARCHAR(100),
    equipment_required VARCHAR(100),
    difficulty ENUM('easy', 'medium', 'hard')
) ENGINE=InnoDB;

CREATE TABLE SessionExercise (
    session_id INT NOT NULL,
    exercise_id INT NOT NULL,
    planned_sets INT,
    planned_reps INT,
    rest_duration INT,
    PRIMARY KEY (session_id, exercise_id),
    CONSTRAINT fk_se_session
        FOREIGN KEY (session_id) REFERENCES WorkoutSession(session_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT fk_se_exercise
        FOREIGN KEY (exercise_id) REFERENCES Exercise(exercise_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
) ENGINE=InnoDB;

CREATE TABLE PerformanceLog (
    athlete_id INT NOT NULL,
    session_id INT NOT NULL,
    exercise_id INT NOT NULL,
    completed_sets INT,
    completed_reps INT,
    weight_used DECIMAL(6,2),
    perceived_exertion INT,
    log_time DATETIME DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (athlete_id, session_id, exercise_id),
    CONSTRAINT fk_log_athlete
        FOREIGN KEY (athlete_id) REFERENCES Athlete(athlete_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT fk_log_session_exercise
        FOREIGN KEY (session_id, exercise_id)
        REFERENCES SessionExercise(session_id, exercise_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
) ENGINE=InnoDB;

CREATE TABLE BodyMeasurement (
    athlete_id INT,
    measurement_date DATE,
    height DECIMAL(5,2),
    weight DECIMAL(5,2),
```

```
    body_fat_percentage DECIMAL(5,2),
    muscle_mass DECIMAL(5,2),
    bmi DECIMAL(5,2),
    PRIMARY KEY (athlete_id, measurement_date),
    CONSTRAINT fk_body_athlete
        FOREIGN KEY (athlete_id) REFERENCES Athlete(athlete_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
) ENGINE=InnoDB;

CREATE TABLE MedicalAssessment (
    athlete_id INT,
    medical_id INT,
    assessment_date DATE,
    assessment_type VARCHAR(100),
    notes TEXT,
    clearance_status ENUM('cleared', 'restricted', 'not_cleared'),
    PRIMARY KEY (athlete_id, medical_id, assessment_date),
    CONSTRAINT fk_ma_athlete
        FOREIGN KEY (athlete_id) REFERENCES Athlete(athlete_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT fk_ma_medical
        FOREIGN KEY (medical_id) REFERENCES Medical(medical_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
) ENGINE=InnoDB;

CREATE TABLE TrainerFeedback (
    trainer_id INT,
    athlete_id INT,
    session_id INT,
    rating INT CHECK (rating BETWEEN 1 AND 5),
    comments TEXT,
    PRIMARY KEY (trainer_id, athlete_id, session_id),
    CONSTRAINT fk_feedback_trainer
        FOREIGN KEY (trainer_id) REFERENCES Trainer(trainer_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT fk_feedback_athlete
        FOREIGN KEY (athlete_id) REFERENCES Athlete(athlete_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT fk_feedback_session
        FOREIGN KEY (session_id) REFERENCES WorkoutSession(session_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
) ENGINE=InnoDB;
```

```
CREATE OR REPLACE VIEW v_log_enriched
AS
  SELECT PL.athlete_id,
         PL.session_id,
         WS.program_id,
         WS.session_date,
         PL.exercise_id,
         E.exercise_name,
         E.type,
         E.equipment_required,
         E.difficulty,
         PL.completed_sets,
         PL.completed_reps,
         PL.weight_used,
         PL.perceived_exertion,
         PL.log_time
  FROM   PerformanceLog PL
         JOIN WorkoutSession WS
           ON WS.session_id = PL.session_id
         JOIN Exercise E
           ON E.exercise_id = PL.exercise_id;
```

Data Sources

The database is populated with fake data generated by Python's Faker library:

- There are 100 users, 60 of which are athletes and 40 of which are staff. Of the staff, there are 15 trainers and 25 medical staff.
- For the purposes of testing, every user's password is generated as a hash of "123456", though the database is capable of storing other hashes normally.
- There are 20 training programs and 30 exercises in the database by default.
- Every athlete has 1 to 3 program enrollments for the default programs.
- Every training program has 10 workout sessions and every workout session has 4 to 8 exercises by default.
- Every athlete has 2 to 5 body measurements and 1 to 3 medical reports by default.
- There are at most 200 tuples of training feedback by default.

Advanced SQL Queries

Query 1:

```
WITH x AS (
        SELECT
            athlete_id,
            exercise_id,
            exercise_name,
            weight_used,
            completed_sets,
            completed_reps,
            perceived_exertion,
            log_time,
            ROW_NUMBER() OVER (
```

```
        PARTITION BY athlete_id, exercise_id
        ORDER BY log_time DESC
        ) AS rn
    FROM v_log_enriched
    )
    SELECT *
    FROM x
    WHERE rn = 1 AND athlete_id = %s
    ORDER BY log_time DESC;
```

What it does (brief): Uses a window function (ROW_NUMBER) to return only the latest log row per *(athlete, exercise)* based on log_time.

Why it is useful: Trainers often want a current snapshot of each athlete's latest performance per exercise (latest weight/reps/sets/RPE). This is exactly the "trainer as power-user" idea.

Program Integration: Training Progress → Last training Info

```
Query 2:
SELECT
        pl.athlete_id,
        pl.session_id,
        ws.session_date,
         ROUND(100 * SUM(pl.completed_sets) / NULLIF(SUM(se.planned_sets), 0),
2) AS percentage_sets_done,
         ROUND(100 * SUM(pl.completed_reps) / NULLIF(SUM(se.planned_reps), 0),
2) AS percentage_reps_done,
                                ROUND(AVG(pl.perceived_exertion),    2)    AS
average_rate_of_perceived_exertion
    FROM PerformanceLog pl
    JOIN SessionExercise se
    ON se.session_id = pl.session_id AND se.exercise_id = pl.exercise_id
    JOIN WorkoutSession ws
    ON ws.session_id = pl.session_id
    WHERE athlete_id = %s
    GROUP BY pl.athlete_id, pl.session_id, ws.session_date
    ORDER BY ws.session_date DESC, pl.athlete_id;
```

What it does: Joins planned session exercises (SessionExercise) with actual logs (PerformanceLog) and computes completion % for sets/reps + average RPE per session.

Why useful in this project: It directly supports "adherence analysis" (planned vs done), which is a key analytic requirement for training tracking apps. It also gives coaches a quick read on effort (RPE) alongside adherence.

Program Integration: Training Progress → Session Adherence

```
Query 3:
WITH vol AS (
    SELECT
        athlete_id,
        exercise_id,
        exercise_name,
         ROUND(SUM(completed_sets * completed_reps * IFNULL(weight_used, 0)),
2) AS total_volume
    FROM v_log_enriched
    GROUP BY athlete_id, exercise_id, exercise_name
    ),
    ranked AS (
    SELECT
        *,
        DENSE_RANK() OVER (
        PARTITION BY athlete_id
        ORDER BY total_volume DESC
        ) AS rnk
    FROM vol
    )
    SELECT athlete_id, exercise_id, exercise_name, total_volume, rnk
    FROM ranked
    WHERE rnk <= 3 AND athlete_id = %s
    ORDER BY athlete_id, rnk, total_volume DESC;
```

What it does (brief): Aggregates total training volume per exercise and athlete, then uses DENSE_RANK to return each athlete's top 3 volume exercises.

Why useful in this project: Great for understanding an athlete's training focus, identifying imbalances, and generating "insight cards" (e.g., athlete's strongest / most-trained lifts).

Program Integration: Training progress → Top 3 Exercises

```
Query 4:
SELECT
            tp.program_id,
            tp.program_name,
            tp.start_date,
            tp.end_date
        FROM trainingprogram tp
        WHERE tp.program_id NOT IN (
            SELECT program_id
            FROM programenrollment
            WHERE athlete_id = %s
        )
        ORDER BY tp.start_date DESC
```

What it does: This query retrieves all training programs that a given athlete is not currently enrolled in. It uses a subquery with NOT IN to exclude programs already present in ProgramEnrollment for that athlete.

Why it is useful in this project: It prevents duplicate enrollments in the same training program and ensures athletes only see relevant and available programs.

Program Integration: Enrollment → Enroll in Program

```
Query 5:
WITH per_prog AS (
            SELECT
                ws.program_id,
                pl.athlete_id,
                COUNT(DISTINCT pl.session_id) AS logged_sessions,
                ROUND(AVG(pl.perceived_exertion), 2) AS avg_rpe
            FROM PerformanceLog pl
            JOIN WorkoutSession ws ON ws.session_id = pl.session_id
            GROUP BY ws.program_id, pl.athlete_id
        ),
        ranked AS (
            SELECT
                *,
                RANK() OVER (
                    PARTITION BY program_id
                    ORDER BY logged_sessions DESC, avg_rpe DESC
                ) AS rnk
            FROM per_prog
        )
        SELECT
            r.program_id,
            tp.program_name,
            r.athlete_id,
            CONCAT(u.first_name, ' ', u.last_name) AS athlete_name,
            r.logged_sessions,
            r.avg_rpe,
            r.rnk
        FROM ranked r
        JOIN TrainingProgram tp ON tp.program_id = r.program_id
        JOIN User u ON u.user_id = r.athlete_id
        WHERE r.rnk <= 5 AND tp.created_by_trainer = %s
        ORDER BY r.program_id, r.rnk, r.athlete_id
```

What it does: This query builds a **leaderboard for each training program created by a specific trainer**:

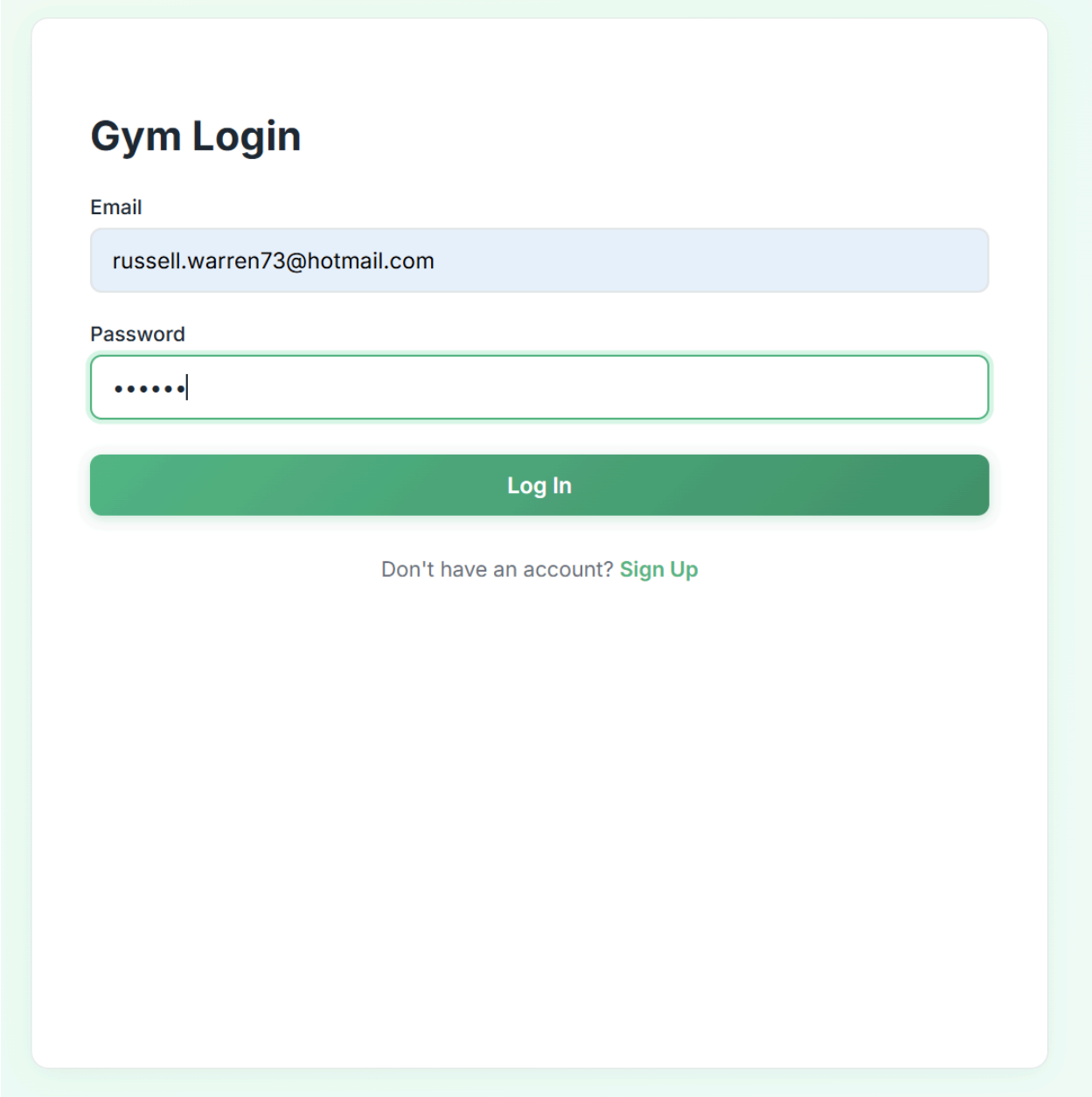
1. `per_prog` **CTE:** For each *(program, athlete)* it calculates
    - `logged_sessions`: how many distinct sessions the athlete logged

    - `avg_rpe`: the athlete's average perceived exertion (RPE) across those logs
2. `ranked` **CTE:** Uses a **window ranking (RANK)** to rank athletes *within each program* by:
    - most `logged_sessions` first (consistency)
    - then higher `avg_rpe` (effort/intensity) as tie-breaker
3. Final SELECT joins `TrainingProgram` and `User` to display **program name** and **athlete full name**, filters to **Top 5** per program, and restricts to **programs owned by the trainer** (`created_by_trainer = %s`).

Why it is useful in this project: Gives trainers an overview of who is most consistent in each of their programs. It also combines quantity (attendance/logged sessions) with quality/intensity (avg RPE). It helps turning logs into actionable coaching insights.

Program Integration: Training Program Management → Program Leaderboards (Top 5 Athletes)

Screenshots



Gym Sign-up

First Name

Enter your first name:

Last Name

Enter your last name:

Email

Enter email:

Password

Enter password:

Phone (format: 123-123-1234)

Gender

Male

Date of Birth

dd/mm/yyyy

Select Role

Athlete

Sports Branch

e.g. Basketball

Sign Up

Already have an account? Log In

Welcome, Russell!

Sign Out

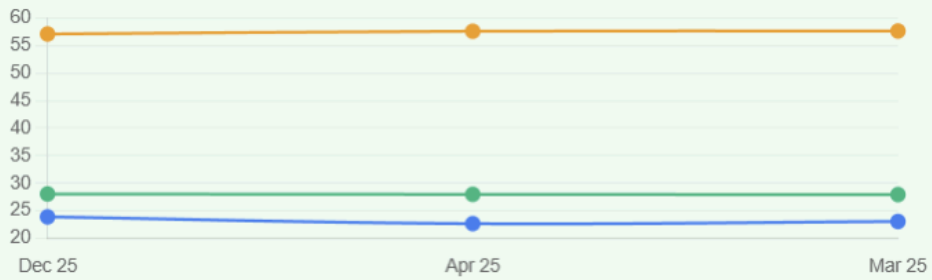
Progress | Management

Training Progress

Athlete

42, Anthony Cooper

Body Measurements



Date	Height (cm)	Weight (kg)	Body Fat %	Muscle Mass (kg)	BMI
Sat, 15 Mar 2025	179.57	90.06	23.04	57.64	27.93
Fri, 25 Apr 2025	179.89	90.50	22.65	57.60	27.97
Thu, 18 Dec 2025	179.30	90.13	23.88	57.10	28.04

Medical Assessment

Doctor	Date	Type	Notes	Clearance
			Hit keep brother short provide.	

Doctor	Date	Type	Notes	Clearance
Victoria Sims	Fri, 03 Oct 2025	Injury Assessment	Hit keep brother short provide. Whom summer bring dream support. Myself work reveal responsibility receive agree. Safe determine say close white since. Age true dinner turn I spend affect. Movement speech skill ability especially win result. Senior central although page husband.	restricted

Top 3 Exercises

Ranking	Exercise	Total Volume (Set * Rep * Weight (kg))
1	T-Bar Row	6115.20
2	Face Pulls	5955.69
3	Pistol Squat	4870.08

Last Training Info

Exercise	Weight (kg) Used	Sets	Reps	Perceived Exertion	Date
T-Bar Row	101.92	5	12	8	Sat, 17 Jan 2026 15:41:47 GMT
Pistol Squat	101.46	3	16	5	Sun, 21 Dec 2025 14:22:09 GMT
Battle Ropes	15.95	3	7	10	Sat, 20 Dec 2025 14:13:36 GMT

Session Adherence

Set Completion (%)	Rep Completion (%)	Average Perceived Exertion	Date
100.00	114.29	5.00	Mon, 22 Dec 2025
100.00	100.00	5.00	Wed, 19 Nov 2025
80.00	83.33	5.00	Tue, 11 Nov 2025
75.00	87.50	10.00	Sun, 26 Oct 2025
75.00	92.86	10.00	Wed, 06 Aug 2025

Training Program Management

Create New Program

Add Workout Session

Add Trainer Feedback

View Leaderboard

Create New Program

Program Name

Difficulty Level

Beginner

Goal

Start Date

dd/mm/yyyy

End Date

dd/mm/yyyy

Create

Training Program Management

Create New Program

Add Workout Session

Add Trainer Feedback

View Leaderboard

Add Workout Session

Select Program

Select a program

Session Date

dd/mm/yyyy

Duration (minutes)

Intensity Level

Low

Add Session

Welcome, Russell!

Sign Out

Progress | Management

Training Program Management

Create New Program

Add Workout Session

Add Trainer Feedback

View Leaderboard

Add Trainer Feedback

Select Athlete

Select an athlete

Select Workout Session

Select a session

Rating

1 - Poor

Feedback Notes

Submit Feedback

Welcome, Russell!

Sign Out

Progress | Management

Training Program Management

Create New Program

Add Workout Session

Add Trainer Feedback

View Leaderboards

Program Leaderboards (Top 5 Athletes)

Strength Building Phase

Rank	Athlete	Sessions Logged	Avg RPE
1	Adam Rhodes	4	8.50
2	Shelly Anderson	2	8.50
3	Paul Daniel	2	7.00
4	Alexander Robles	2	6.50
5	Karen Flores	2	6.00

Welcome, Theresa!

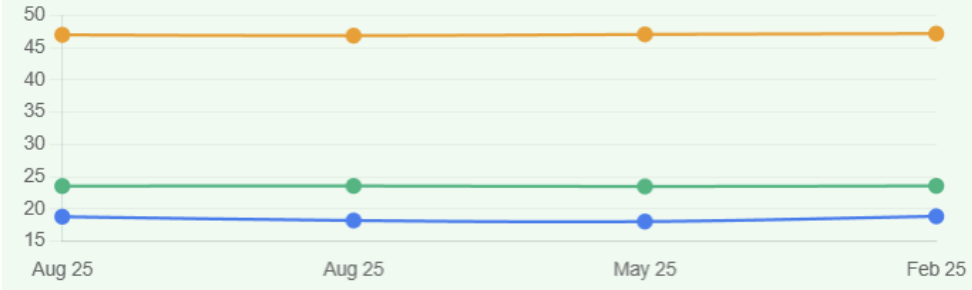
Sign Out

Progress | Enrollment

Training Progress

Athlete

Body Measurements



Date	Height (cm)	Weight (kg)	Body Fat %	Muscle Mass (kg)	BMI
Wed, 05 Feb 2025	191.38	86.34	18.85	47.19	23.57
Fri, 09 May 2025	191.61	86.22	18.04	47.07	23.48
Sat, 09 Aug 2025	191.53	86.42	18.20	46.87	23.56
Sat, 16 Aug 2025	191.36	86.19	18.79	47.00	23.54

Medical Assessment

Doctor	Date	Type	Notes	Clearance
John Ho	Tue, 09 Sep 2025	Recovery Assessment	Card pretty consider two since activity. Report everybody care of practice task skill most. Share general agent off film hair. Plan job	cleared

Welcome, Theresa!

Sign Out

Progress | Enrollment

Enrollment

Available Programs

Select a program to enroll

Select a program to enroll

Enroll in Program

My Enrolled Programs

Enrolled Programs

Select a program

Workout Sessions

Session Date	Duration (min)

Welcome, Theresa!

Sign Out

Progress | Enrollment

Enrollment

Available Programs

Select a program to enroll

Speed and Agility: (Mon, 05 Jan 2026 - Mon, 20 Apr 2026)

Enroll in Program

My Enrolled Programs

Enrolled Programs

Endurance Training: (Fri, 25 Jul 2025 - Sun, 16 Nov 2025)

Workout Sessions

Session Date	Duration (min)
Fri, 03 Oct 2025	87
Sun, 07 Sep 2025	50
Sat, 23 Aug 2025	119
Wed, 06 Aug 2025	51
Sat, 19 Jul 2025	40
Fri, 18 Jul 2025	38

Medical Assessment

Athlete

90, Alexander Robles

Body Measurements

Date	Height (cm)	Weight (kg)	Body Fat %	Muscle Mass (kg)	BMI
Sun, 19 Jan 2025	167.95	69.58	16.89	57.69	24.67
Tue, 13 May 2025	168.06	70.52	16.25	56.70	24.97

Medical Assessment

Doctor	Date	Type	Notes	Clearance
Steven Walton	Tue, 04 Nov 2025	Clearance Check	Bad age enjoy event realize positive think. Rather area cause deep his. List they interesting street subject. Her day owner huge. Stage technology discover history stage heavy baby meeting. Fact food off walk. Provide player least cover think follow indicate. Eat break compare.	restricted

New Medical Exam

Assessment Type:

Recovery Assessment

Examination Notes

Clearance Status

Cleared

Submit Examination