Is now working on the trainer dashboard, where each trainer should only see and submit data related to their own Branch and Gender. The login flow must dynamically redirect trainers and store session-based Branch and Gender info. Five APIs will be updated to enforce these filters.

Is building a 'fitness-test' webpage where students enter their TR number to load personal data from the `TestMaster` table. The form then collects fitness inputs (weight, height, waist, hips, neck, push-ups, etc.), calculates health metrics (BMI, body fat %, BMR, VO2 max, etc.), displays a report, and saves the results into a new table `TestRecords` with relevant columns.

Is building a universal gym management system for 4 branches and 2 genders, with separate dashboards for Trainers and Admins but shared backend logic. The project includes phased development: database redesign, file structure setup, secure login flow, role-based routing, shared backend APIs, modular frontend scripts, and deployment for controlled access.

SQL Server database contains the following tables and structures:

1. Master (TR, Name, Darajah, Goal, Slot)

2. AttendanceWeek (WeekID, WeekStartDate, WeekEndDate)

3. Attendance (AttendanceID, TR, WeekID, IsPresent, CreatedAt)

4. PassBank (Username, Password)

5. Machines (MachineID, MachineName, TypeName, InputType)

6. WorkoutLog (LogID, TR, MachineID, Value, LogDate)

Has 3 user roles in their gym management system: Student, Staff/Admin, and Trainer. Each role has separate login APIs and dashboards. Students log in using TR number (matched in Master table), while staff and trainers log in using credentials in the PassBank table. Each login redirects users to a branch- and gender-specific dashboard. Student login HTML page is already created.

Universal staff dashboard is named 'dashboard.html' and includes tabs for Student Entry, Student Record, Track Progress, and Attendance Summary. It supports adding students, importing from CSV, filtering by Darajah and Slot, deleting entries, and viewing workout logs and weekly attendance.

Universal staff dashboard HTML is named 'dashboard.html'. It includes tabs for Student Entry, Student Record, Track Progress, and Attendance Summary. It supports adding students, bulk CSV import, rendering student records with filters, deleting entries, loading weekly attendance, and viewing workout progress.

Wants to restrict staff actions based on their assigned Branch and Gender. Student entries should inherit Branch and Gender from the logged-in staff. APIs for student management, attendance, and workout logs must enforce these constraints. The `Master`, `Attendance`, `WorkoutLog`, and `PassBank` tables include Branch and Gender columns.

Is learning determinants and wants to rigorously understand the concept by solving a series of problems with guidance.

Wants to build an official staff dashboard view with a single table showing all workout logs, including TR, Name, Machine, Type, Value, and Date. The table should support filtering, sorting, and exporting to Excel.

SQL Server database has six tables: Master, AttendanceWeek, Attendance, PassBank, Machines, and WorkoutLog. Each table's columns are:

- Master: TR, Name, Darajah, Goal, Slot

- AttendanceWeek: WeekID, WeekStartDate, WeekEndDate

- Attendance: AttendanceID, TR, WeekID, IsPresent, CreatedAt

- PassBank: Username, Password

- Machines: MachineID, MachineName, TypeName

- WorkoutLog: LogID, TR, MachineID, Reps, WeightUsed, WorkoutDate.

Wants to redesign the workout logging system so that students only enter one type of input per machine (Reps, Weight, or Duration) based on the machine’s use. They plan to remove or repurpose the `WeightUsed` column and keep only a unified metric (e.g., Reps). The frontend should display machine types as ribbons, each with clickable machine boxes. When clicked, a counter appears (initially hidden), allowing the student to increment and then submit. User wants a fully functioning, clean system even if it means restructuring tables.

Is redesigning the workout logging system from scratch. They are dropping the `WeightUsed` column from `WorkoutLog`, starting fresh with new APIs and frontend. Machines are: 1. Treadmill (Cardio) – Duration, 2. Cycling (Cardio) – Duration, 3. Elliptical (Cardio) – Duration, 4. Bench Press (Strength) – Reps, 5. Leg Press (Strength) – Reps, 6. Lat Pulldown (Strength) – Reps, 7. Stretching Station (Flexibility) – Duration. The frontend will use ribbons for machine types and interactive machine boxes that display counters only upon selection.

Is redesigning the student workout logging interface to show machine types as ribbons with machine cards below them. Each machine card will show a counter (Value) only when clicked.

In the `WorkoutLog` table, the `WeightUsed` column has been dropped and the `Reps` column has been renamed to `Value`.

Prefers each machine workout entry to have its own individual submit button (not a single global one). They want a clean frontend with Type ribbons, machine cards that reveal counters and submit buttons upon click, and each submission should send only that machine's data.

Has a `student-dashboard.html` with an existing script that handles student authentication, attendance loading, and weekly tips. They want to add the new machine logging frontend without breaking existing functionality.

Wants to add a new tab in the student dashboard to display workout history using a Bootstrap table. The plan includes creating a GET API per student TR and the corresponding frontend script.

`WorkoutLog` table has the column `LogDate` instead of `WorkoutDate`.

Wants to recreate the entire GET and POST system for their gym workout tracking app from scratch, including all tables, APIs, frontend HTML/CSS/JS, with properly working date handling and no server-side 500 errors.

Has three main APIs: two GET routes for machines and one POST route for workout logs. They also have a GET route for fetching student workout summaries. They want to review and clean up these APIs before moving to frontend integration.

Project now involves supporting a gym system across four branches (Marol, Surat, Nairobi, Karachi), each with separate Male and Female panels, totaling 8 admin panels. Each admin must only see data relevant to their branch and gender. The system will need a universal Master table with unique student IDs ("TR") as primary keys, and data access partitioned based on Branch and Gender.

Wants to build a real workout progress system for students using a point-based model. Each workout type (e.g., Leg Day) includes exercises (e.g., squats, calve press), each with reps, which contribute to a cumulative point score. This point system will increase a student's progress bar. The staff will use this system to track student performance across the year.

Is implementing a points-based workout progress system where students earn points for exercises within workout types (e.g., Leg Day), and staff can track student progress over time. The system includes logging exercises, reps, and calculating cumulative points to drive a progress bar visualization in the student dashboard.

Is building a gym attendance and dashboard system for students, with three webpages: staff-dashboard (index.html), trainer-dashboard, and student-dashboard. They are using HTML, CSS, Node.js, and Microsoft SQL Server (via Somee.com). There are three SQL tables: Master, AttendanceWeek, and Attendance. They use one `server.js` backend file, which works fine for index.html but causes JSON parsing errors for the other dashboards.

Is building a staff dashboard for gym membership data. They are using Microsoft SQL Server 2022 locally, with a database named 'fittracker' and a table named 'Master' containing columns 'TR', 'Name', 'Darajah', 'Goal', and 'Slot'. They previously used MySQL but are transitioning to SQL Server.

Is using a staff dashboard HTML file that interacts with a SQL Server database ('fittracker') and the 'Master' table. They want to remove all MySQL-related code and integrate real-time SQL-based interactions, including student record display and bulk CSV import.

Connects to their SQL Server database using Windows Authentication, not SQL login.

Wants their frontend dashboard to directly communicate with their local SQL Server 2022 database (using Windows Authentication) without relying on localhost:3000. They want CSV import and display of student records from the 'fittracker.dbo.Master' table.

CREATE TABLE TestMaster (

TR INT PRIMARY KEY,

ITS INT,

Darajah INT,

Age INT,

Name NVARCHAR(100),

Hizb NVARCHAR(50),

Class NVARCHAR(20),

House NVARCHAR(50),

Check18 NVARCHAR(10),

Email NVARCHAR(100),

DOB DATE

);

CREATE TABLE TrainingPlan (

PlanID INT IDENTITY(1,1) PRIMARY KEY,

TR INT NOT NULL,

BodyParts NVARCHAR(100) NOT NULL, -- Comma-separated like 'Chest,Triceps'

CreatedAt DATETIME NOT NULL DEFAULT GETDATE(),

Branch NVARCHAR(50) NOT NULL,

Gender NVARCHAR(10) NOT NULL,

CONSTRAINT FK\_TrainingPlan\_Master FOREIGN KEY (TR) REFERENCES Master(TR)

);