Gym Management System — Introduction 🏋️

In this project, students will build a **Gym Management System** that helps organize gym members, trainers, and workout plans.
The system includes a backend (Django + MySQL) and a frontend (Angular) for managing data and displaying it in a user-friendly way

Use VScode with Copilot

We will divide the work plan into three parts: DB, API and FrontEnd.

DATABASE

1) DB Tables:

We will have 3 tables:
User: in this table we will store all the gym members and staff (trainers and admins)

Plans: this tables will include all the plans for all the members

Machines: this table will include all the gym machines

2) Tables relationships:

Users → Plans:
Each trainee can have **one or more workout plans**.

Trainers → Trainees:
A trainer can have **multiple trainees**, while each trainee has **one trainer**.

Machines → Plans:
Plans reference the machines used by storing their IDs (as a comma-separated list).

3) Tables illustration:

- **Users table:**

```
id CHAR(36) PRIMARY KEY,
name VARCHAR(50) NOT NULL,
lastname VARCHAR(50) NOT NULL,
email VARCHAR(100) UNIQUE NOT NULL,
phoneNumber VARCHAR(20),
type ENUM('trainer', 'trainee', 'admin') NOT NULL,
trainerId CHAR(36) DEFAULT NULL,
active BOOLEAN DEFAULT TRUE,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
FOREIGN KEY (trainerId) REFERENCES users(id)
```

- **Machines Table**

```
id CHAR(36) PRIMARY KEY,
code VARCHAR(10) NOT NULL,
name VARCHAR(100) NOT NULL,
description TEXT
```

- **Plans Table**

```
id CHAR(36) PRIMARY KEY,
traineeId CHAR(36) NOT NULL,
description VARCHAR(100),
machines VARCHAR(255),
days VARCHAR(100),
sets INT DEFAULT 3,
reps INT DEFAULT 15,
durationMinutes INT DEFAULT NULL,
createdAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
FOREIGN KEY (traineeId) REFERENCES users(id)
```

API

Now we will display the needed API routes for our app.

1) Authentication

Route	Method	Purpose
`/api/auth/login`	POST	Log in a user (returns a token or session info)
`/api/auth/register`	POST	Create a new user (trainee, trainer, or admin)
`/api/auth/logout`	POST	Log out the user (invalidate token/session)
`/api/auth/me`	GET	Get info about the currently logged-in user using token/session

2) Users

Route	Method	Purpose
`/api/users`	GET	Get a list of all users
`/api/users/:id`	GET	Get details of a single user by ID
`/api/users`	POST	Create a new user (admin adds a trainee/trainer/admin)
`/api/users/:id`	PUT	Update user info
`/api/users/:id`	DELETE	Delete a user (soft delete by setting `active = false` is recommended)

The **/api/users** route should have optional query params to enable filtering and sorting the users by:

- active/inactive
- trainer
- asc order
- desc order

3) Plans

Route	Method	Purpose
`/api/plans`	GET	Get all plans
`/api/plans/:id`	GET	Get a single plan by ID
`/api/plans`	POST	Create a new plan
`/api/plans/:id`	PUT	Update an existing plan
`/api/plans/:id`	DELETE	Delete a plan

The **/api/plans** route should have query params to enable filtering the plans by:

- userId (we want to get all the plans for a specific user)

4) Machines

Route	Method	Purpose
`/api/machines`	GET	Get all machines
`/api/machines/:id`	GET	Get a single machine by ID
`/api/machines`	POST	Create a new machine
`/api/machines/:id`	PUT	Update an existing machine
`/api/machines/:id`	DELETE	Delete a machine

The /api/machines should have query params to enable filtering by:

- description

FRONTEND

Functionality Requirements

Your app must allow the following actions:

1. **Authentication**
    - Login functionality.
    - Display current user info after login.
2. **User Management**
    - Admin can add new users.
    - Admin can edit user information.
    - Admin can deactivate or delete users.
    - Display users filtered by type (trainee, trainer, admin).
3. **Machine Management**
    - Add, edit, delete machines.
    - Display list of all machines.
4. **Plan Management**
    - Add, edit, delete workout plans.
    - Assign plans to trainees.
    - Display plans for a specific trainee.
    - Include machines, workout days, sets, reps, and duration.

Components:

You can design and style your app as you wish as long as it has the mentioned functionalities.

My recommendations are:

Components depends on the user type:

1) Regular member: will have the following:
    - **landing page**- home: contains today's plan for example: today's date and what are the workouts for today.
    - **Profile**: a page that displays the details of the member
    - **My trainer**: a page that contains the details of the trainer
    - **Contact us**: a page that displays the details of the gym: working hours, contact details

2) Trainer member: will have the following:
    - **Landing page**: hope: a table that displays all the trainees members: name, lastname, Status(active/inactive) and a button to display a specific trainee- the table should have ability to sort by name in asc and desc order.
    - **Specific trainee details**: a page that display a specific trainee details when clicking on it in the trainees table
    - **Profile**: a page that displays the details of the trainee with the ability to edit his details

3) Admin member: will have the following:
    - **Landing page**- dashboard: it shows statics (you can use charts for better design) about members activity- how many members are active for example, how many members are registered etc…(be creative)
    - **Trainers:** a page that contains a table of all the trainers (name, lastname and status) when clicking on a trainer name should display all the details of the trainer including a table of their trainees with the ability to activate/inactivate him and edit his details (the table should have the ability to filter by status and sort by name asc/desc)
    - **Members:** a page that contains a table of all the members in the gym (name, lastname and status) when clicking on a specific member it should display all the details about it including the name of his trainer (with the ability to click on the trainer name and it will take you a the trainer's details page- use same components, don't duplicate components!!!), (the table should have the ability to filter by status and sort by name asc/desc)
    - **Profile**: a profile page that shows the details of the admin, and the ability to add a new member/trainer/admin with the ability to edit his own details.
    - **Machines:** displays all the machines in the gym and the ability to add/delete new machines.

NOTES:

- Do not duplicate components, for example: create one generic table component and use it each time you need to implement a table use, one profile page and use it for each member type with small changes in each profile type..
- Use **pipes**, **forms** and **forms validations** as learned in the course.
- Use **JWT** (you can search in google/youtube and learn it)

use an interceptor and auth guard as well.

- You can use any library for style like MaterialUI and bootstrap and you can download a full template from google and use it