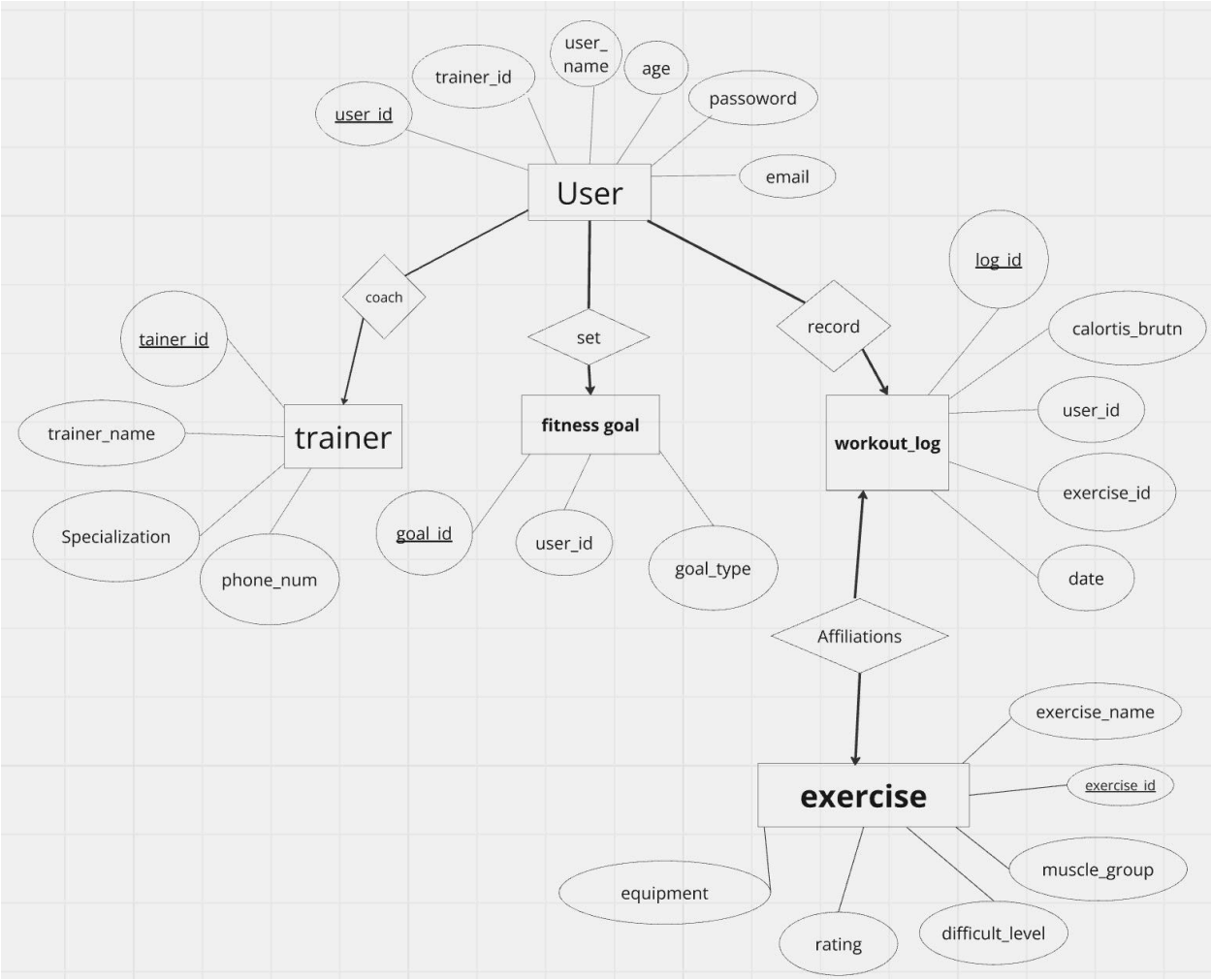
ER Diagram



Description

Entities and Assumptions:

1. User (Accounts):
    a. Attributes: user_id (primary key), trainer_id (foreign key), user_name, age, password, email.
    b. Assumptions: the user entity represents an individual who interacts with the system, whether to log exercises, set goals, or consult a trainer.
    c. Entity vs. Attribute: the user is a fundamental component of the system, so it is modeled as an entity to allow for detailed relationships with other entities. It

cannot be treated as an attribute of another entity because it is the central actor in the system.

2. Trainer:
    a. Attributes: trainer_id (primary key), trainer_name, specialization, phone_num.
    b. Assumptions: trainers provide guidance to users, the specialization attribute helps in matching trainers to user needs based on specific fitness goals or exercise routines.
    c. Entity vs. Attribute: trainers are modeled as a separate entity because they interact with multiple users, which requires managing relationships independently. If thighs info were stored as an attribute in the user entity, it would be redundant for trainers who are linked to multiple users.
3. Workout_log:
    a. Attribute: log_id (primary key), user_id (foreign key), exercise_id (foreign key), date, calories_burnt.
    b. This entity tracks each individual workout session for a user. Each workout log records exercises, duration and calories burnt. A user can have multiple workout logs, reflecting different workouts over time.
    c. Entity vs. Attribute: a workout log is modeled as an entity because it contains detailed data related to a user's performance in specific exercises over time. It wouldn't make sense as an attribute of the user, as each user can have numerous workout sessions.
4. Exercise:
    a. Attribute: exercise_id (primary key), exercise_name, muscle_group, difficulty_level, rating, equipment.
    b. Assumptions: The exercise entity stores information about exercises, including what muscle group it targets and its difficulty level. This allows users to track what kind of exercises they are doing and how challenging they are.
    c. Entity vs Attribute: Exercise is modeled as an entity because it has multiple important attributes (name, difficulty level, etc.) and needs to be linked to workout logs independently. Making exercise just an attribute of the workout log would prevent its reuse across different workout logs.

5. Fitness_Goal:
    a. Attribute: goal_id (primary key), user_id (foreign key), goal_type.
    b. Assumptions: Each user can have multiple fitness goals, such as weight loss, muscle gain, or endurance improvement. These goals are used to track user progress over time.
    c. Entity vs Attribute: Fitness goals are modeled as a separate entity because users can have multiple distinct goals over time. If this information were stored directly within the user entity, it would limit the system's ability to track and update goals independently.

Normalization

Our designed schema conforms to the Third Normal Form (3NF). It ensures that all data elements within the tables are not only uniquely identified by the primary key but also remain mutually independent, with no additional functional dependencies existing among them.
Schema Analysis:

1. user table

   Functional Dependencies: id → trainer_id, user_name, age, password, email
    - Candidate Key: id
    - Non-Key Attributes: trainer_id, user_name, age, password, email

   All non-key attributes are fully functionally dependent on the primary key id. There are no transitive dependencies among the non-key attributes. Therefore, the user table complies with 3NF.
2. trainer Table

   Functional Dependencies: id → trainer_name, specialization, phone_num
    - Candidate Key: id
    - Non-Key Attributes: trainer_name, specialization, phone_num

   All non-key attributes are fully functionally dependent on the primary key id. There are no transitive dependencies. Thus, the trainer table complies with 3NF.
3. workout_log Table

   Functional Dependencies: id → user_id, exercise_id, date, calories_burnt
    - Candidate Key: id

- Non-Key Attributes: user_id, exercise_id, date, calories_burnt

All non-key attributes are fully functionally dependent on the primary key id. There are no transitive dependencies. Therefore, the workout_log table complies with 3NF.

4. fitness_goal Table

   Functional Dependencies: id → user_id, goal_type

   - Candidate Key: id
   - Non-Key Attributes: user_id, goal_type

   All non-key attributes are fully functionally dependent on the primary key id. There are no transitive dependencies. Hence, the fitness_goal table complies with 3NF.

5. exercise Table

   Functional Dependencies: id → exercise_name, muscle_group, difficulty_level, rating, equipment

   - Candidate Key: id
   - Non-Key Attributes: exercise_name, muscle_group, difficulty_level, rating, equipment

   All non-key attributes are fully functionally dependent on the primary key id. There are no transitive dependencies. Therefore, the exercise table complies with 3NF.

Based on the provided schema and the analysis above, all tables comply with the Third Normal Form (3NF). Each table ensures that non-key attributes are fully functionally dependent on the primary key, and there are no transitive dependencies among non-key attributes.

Logical Design

user(id: INT [PK], trainer_id: INT [FK to trainer.id], user_name: VARCHAR(255), age: INT, password: VARCHAR(255), email: VARCHAR(255))

trainer(id: INT [PK], trainer_name: VARCHAR(255), specialization: VARCHAR(255), phone_num: VARCHAR(15))

workout_log(id: INT [PK], user_id: INT [FK to user.id], exercise_id: INT [FK to exercise.id], date: TIMESTAMP, calories_burnt: INT)

fitness_goal(id: INT [PK], user_id: INT [FK to user.id], goal_type: VARCHAR(255))

exercise(id: INT [PK], exercise_name: VARCHAR(255), muscle_group: VARCHAR(255), difficulty_level: VARCHAR(255), rating: FLOAT, equipment: VARCHAR(255))

Revision of Creative Component

1. **Customized workout calendar**: we will ensure it's not just a calendar but an interactive tool that suggests workout adjustments based on user input, like skipped workouts or overachieved goals. For implementation, we will use a simple calendar API for reminders and scheduling.
2. **Goal Achievement Badges**: we will incorporate a basic reward system where users earn badges based on predefined workout milestones, such as completing a set number of workouts or reaching a specific calorie goal. For implementation, we can track user activities in the database and award badges based on these records.
3. **Detailed Fitness Plan Recommender (Revised)**: for a simpler implementation, instead of a complex recommendation system, we can use rule-based logic. Users provide a target calorie expenditure, and based on their input (e.g., height, weight, age), we calculate a basic plan using predetermined exercise templates.

Idea of having more grounded data

After talking with Ti-Chung Cheng, we are planning to connect Youtube videos to our current project. Youtube videos will be provided by us on the app and users have access to different types of videos. Users can choose to store their favorite videos or playlists on our app. Which means we will have a dataset to store links to users' favorite videos or playlists.