# 1. Detailed Schema Overview

## A. Core Data (Shared Definitions & Base Classes)

### 1. BodySegment (from core/models.py)

A TextChoices enumeration that standardizes body segments across the app.
**Key values:**

- `HEAD` ("HD", "Head")
- `TORSO_UPPER` ("TU", "Upper Torso")
- `TORSO_LOWER` ("TL", "Lower Torso")
- `ARM_L_UPPER` ("ALU", "Left Upper Arm")
- `ARM_L_LOWER` ("ALL", "Left Lower Arm")
- `ARM_R_UPPER` ("ARU", "Right Upper Arm")
- `ARM_R_LOWER` ("ARL", "Right Lower Arm")
- `LEG_L_UPPER` ("LLU", "Left Upper Leg")
- `LEG_L_LOWER` ("LLL", "Left Lower Leg")
- `LEG_R_UPPER` ("LRU", "Right Upper Leg")
- `LEG_R_LOWER` ("LRL", "Right Lower Leg")

### 2. BaseTimeStampModel (abstract base in core/models.py)

Provides automatic timestamp fields for creation and updates.
**Fields:**

- `created_at` (DateTime)
- `updated_at` (DateTime)

## B. User Management

### CustomUser (from users/models.py)

Extends Django's `AbstractUser` to add fitness, demographic, and profile details.

**Key Fields:**

- Standard user fields (username, password, etc.)
- `firstName` and `lastName`
- `email` (EmailField)
- `phone`

- `dob` (DateField), `gender`, `age`
- `units` (preferred measurement system)
- `weight` and `height`
- `goals`, `bio`, `profile_picture`
- `role` (using `UserRoles` TextChoices: Free, Premium, Admin)

## C. Workouts & Exercise Tracking

### 1. ExerciseType (from workouts/models.py)

Defines a template for exercises.

**Key Fields:**

- `name`, `description`
- `difficulty_level` (numeric)
- `target_muscles` (JSON list)
- `form_guidelines`
- `video_tutorial_url`

### 2. Workout (from workouts/models.py)

Represents an entire workout session.

**Key Fields:**

- `user` (ForeignKey to CustomUser; optional)
- `title`
- `date_time` (default to now)
- `duration` (DurationField)
- `notes`
- `completed` (Boolean)

### 3. Exercise (from workouts/models.py)

An instance of an exercise performed in a workout.

**Key Fields:**

- `workout` (ForeignKey to Workout)
- `exercise_type` (ForeignKey to ExerciseType)
- `order` (to sequence the exercise)
- `notes`

**4. ExerciseSet (from workouts/models.py)**

A specific set within an exercise.

**Key Fields:**

- `exercise` (ForeignKey to Exercise)
- `set_number` (unique together with exercise)
- `reps` (number of repetitions)
- `weight` (if applicable)
- `duration` (how long the set lasted)
- `rest_after` (rest duration)
- `started_at`, `completed_at` (timestamps)

**Meta Options:**
`ExerciseSet` is ordered by `set_number` and enforces a unique constraint on (`exercise`, `set_number`).

## D. Analytics & Performance Tracking

**1. BodySegmentPosition (from analytics/models.py)**

Stores the 3D position and orientation of a given body segment during an exercise set.

**Key Fields:**

- `exercise_set` (ForeignKey to ExerciseSet from workouts)
- `timestamp` (Float; seconds from set start)
- `segment` (CharField with choices from BodySegment)
- **Position:** `x`, `y`, `z` (normalized coordinates)
- **Orientation:** `pitch`, `yaw`, `roll` (Euler angles in degrees)
- **Relationships:**
    - `angle_to_parent` (angle difference in degrees)
    - `distance_to_parent` (normalized distance)

**Indexes & Ordering:**

- Index on (`exercise_set`, `timestamp`)
- Index on `segment`
- Default ordering by `timestamp`

**2. ExerciseAnalysis (from analytics/models.py)**

Aggregates metrics for an entire exercise set.

**Key Fields:**

- `exercise_set` (OneToOneField to ExerciseSet)
- `segment_ranges` (JSON; max angular ranges per segment)
- `joint_centering` (JSON; alignment scores 0–100)
- `symmetry_score` (Float; overall left/right movement symmetry)
- `fluidity_score` (Float; movement smoothness)

**Additional Methods:**

- `calculate_segment_metrics()`: Placeholder for processing raw position data into metrics.

## E. Notifications & Real-Time Feedback

### 1. Notification (from notifications/models.py)

Represents an in-app notification to users.

**Key Fields:**

- `user` (ForeignKey to CustomUser)
- `title` and `message`
- `created_at`
- `read` (Boolean)
- `notification_type` (Choices: SYSTEM, ACHIEVEMENT, REMINDER, CORRECTION)

**Ordering:**

- Ordered descending by `created_at`

### 2. FormCorrection (from notifications/models.py)

Provides real-time, AI-generated form corrections based on analytics.

**Key Fields:**

- `analysis` (ForeignKey to ExerciseAnalysis)
- `timestamp` (Float; point in the exercise set)
- `segment` (CharField with BodySegment choices)
- `severity` (PositiveSmallIntegerField; choices: Minor, Moderate, Critical)

- `message` (detailed feedback)
- `suggested_cue` (recommended adjustment)

**Ordering:**

- Ordered by `timestamp`

## F. Plans & Structured Workouts

### 1. WorkoutPlan (from plans/models.py)

A structured workout program spanning multiple weeks.

**Key Fields:**

- `name`, `description`
- `creator` (ForeignKey to CustomUser)
- `is_public` (Boolean)
- `difficulty_level`
- `duration_weeks`
- `created_at`

### 2. PlanSubscription (from plans/models.py)

Tracks a user's enrollment in a workout plan.

**Key Fields:**

- `user` (ForeignKey to CustomUser)
- `plan` (ForeignKey to WorkoutPlan)
- `start_date`
- `is_active` (Boolean)
- `current_week`

### 3. PlannedWorkout (from plans/models.py)

A template for a specific workout session within a plan.

**Key Fields:**

- `plan` (ForeignKey to WorkoutPlan)
- `name`, `description`
- `week_number` (which week in the plan)
- `day_of_week` (choices Monday–Sunday)

- `estimated_duration` (optional DurationField)

**Ordering:**

- Ordered by `week_number` then `day_of_week`

**4. PlannedExercise (from plans/models.py)**

An exercise within a planned workout session.

**Key Fields:**

- `planned_workout` (ForeignKey to PlannedWorkout)
- `exercise_type` (ForeignKey to ExerciseType)
- `order` (to sequence within the workout)
- `sets`, `reps` (a range or fixed number)
- `rest_seconds`
- `notes`

**Ordering:**

- Ordered by `order`

## G. Educational Resources

**1. ResourceCategory (from resources/models.py)**

Defines categories for educational content.

**Key Fields:**

- `name`
- `slug` (unique)
- `description` (optional)

**Meta:**

- Verbose plural: “Resource categories”

**2. Article (from resources/models.py)**

Represents educational articles or blog posts.

**Key Fields:**

- `title` and `slug`
- `author` (ForeignKey to CustomUser, with PROTECT on delete)
- `category` (ForeignKey to ResourceCategory, PROTECT)
- `content` and `summary`
- `featured_image` (URLField)
- `published` (Boolean) and `published_date`
- `created_at`, `updated_at`
- `related_exercises` (ManyToManyField with ExerciseType)
  *This links the article to specific exercise types for context.*

**3. Tutorial (from resources/models.py)**

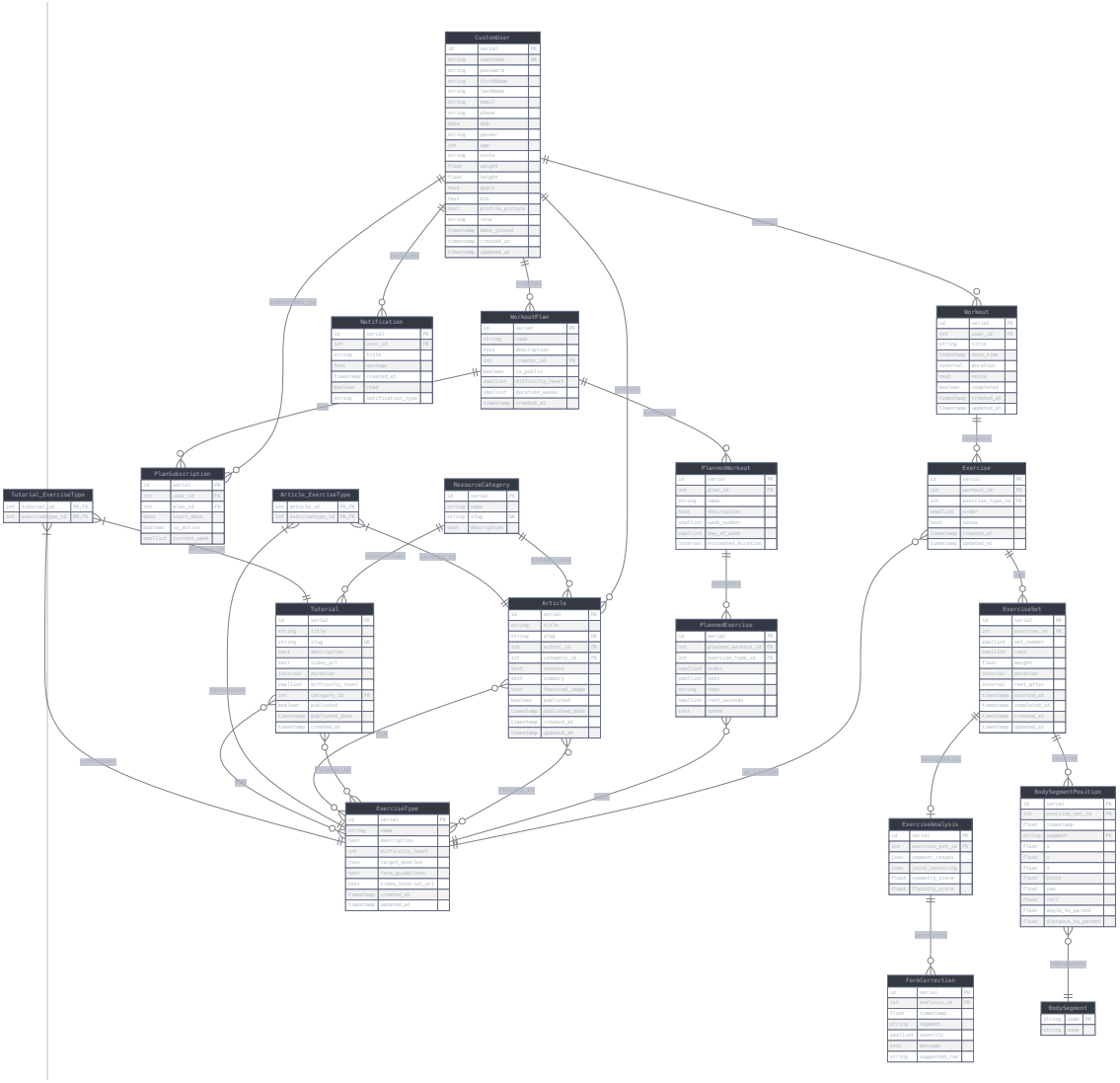
Handles video or media-rich tutorials.

**Key Fields:**

- `title`, `slug`, and `description`
- `video_url`
- `duration` (DurationField)
- `difficulty_level`
- `category` (ForeignKey to ResourceCategory, PROTECT)
- `published` and `published_date`
- `created_at`
- `related_exercises` (ManyToManyField with ExerciseType)

# 2. Entity-Relationship Diagram

Below is an ER diagram showing the major entities and relationships:



# 3. SQL DDL Examples

Below are sample SQL DDL statements for a few key tables. You can expand and adjust these as needed:

```
-- CustomUser (users)
CREATE TABLE CustomUser (
```

```
    id SERIAL PRIMARY KEY,
    username VARCHAR(150) UNIQUE NOT NULL,
    password VARCHAR(128) NOT NULL,
    firstName VARCHAR(150),
    lastName VARCHAR(150),
    email VARCHAR(254),
    phone VARCHAR(20),
    dob DATE,
    gender VARCHAR(20),
    age INTEGER,
    units VARCHAR(20),
    weight FLOAT,
    height FLOAT,
    goals TEXT,
    bio TEXT DEFAULT '',
    profile_picture TEXT DEFAULT '',
    role VARCHAR(10) DEFAULT 'Free',
    -- plus additional fields from AbstractUser as needed
    date_joined TIMESTAMP NOT NULL DEFAULT
CURRENT_TIMESTAMP
);

-- ExerciseType (workouts)
CREATE TABLE ExerciseType (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    difficulty_level INTEGER DEFAULT 1,
    target_muscles JSON NOT NULL DEFAULT '[]',
    form_guidelines TEXT,
    video_tutorial_url TEXT
);

-- Workout (workouts)
CREATE TABLE Workout (
    id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES CustomUser(id) ON DELETE
CASCADE,
    title VARCHAR(100) NOT NULL,
    date_time TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
```

```
    duration INTERVAL,
    notes TEXT,
    completed BOOLEAN DEFAULT FALSE
);

-- Exercise (workouts)
CREATE TABLE Exercise (
    id SERIAL PRIMARY KEY,
    workout_id INTEGER REFERENCES Workout(id) ON DELETE
CASCADE,
    exercise_type_id INTEGER REFERENCES ExerciseType(id) ON
DELETE PROTECT,
    "order" SMALLINT DEFAULT 0,
    notes TEXT
);

-- ExerciseSet (workouts)
CREATE TABLE ExerciseSet (
    id SERIAL PRIMARY KEY,
    exercise_id INTEGER REFERENCES Exercise(id) ON DELETE
CASCADE,
    set_number SMALLINT NOT NULL,
    reps SMALLINT,
    weight FLOAT,
    duration INTERVAL,
    rest_after INTERVAL,
    started_at TIMESTAMP,
    completed_at TIMESTAMP,
    UNIQUE (exercise_id, set_number)
);

-- BodySegmentPosition (analytics)
CREATE TABLE BodySegmentPosition (
    id SERIAL PRIMARY KEY,
    exercise_set_id INTEGER REFERENCES ExerciseSet(id) ON
DELETE CASCADE,
    timestamp FLOAT NOT NULL,
    segment VARCHAR(3) NOT NULL,  -- enforce values via
application logic
    x FLOAT NOT NULL,
```

```
    y FLOAT NOT NULL,
    z FLOAT NOT NULL,
    pitch FLOAT NOT NULL,
    yaw FLOAT NOT NULL,
    roll FLOAT NOT NULL,
    angle_to_parent FLOAT NOT NULL,
    distance_to_parent FLOAT NOT NULL
);
CREATE INDEX idx_bsp_exercise_set_timestamp ON
BodySegmentPosition (exercise_set_id, timestamp);
CREATE INDEX idx_bsp_segment ON BodySegmentPosition
(segment);

-- ExerciseAnalysis (analytics)
CREATE TABLE ExerciseAnalysis (
    id SERIAL PRIMARY KEY,
    exercise_set_id INTEGER UNIQUE REFERENCES
ExerciseSet(id) ON DELETE CASCADE,
    segment_ranges JSON NOT NULL DEFAULT '{}',
    joint_centering JSON NOT NULL DEFAULT '{}',
    symmetry_score FLOAT,
    fluidity_score FLOAT
);

-- Notification (notifications)
CREATE TABLE Notification (
    id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES CustomUser(id) ON DELETE
CASCADE,
    title VARCHAR(100) NOT NULL,
    message TEXT NOT NULL,
    created_at TIMESTAMP NOT NULL DEFAULT
CURRENT_TIMESTAMP,
    read BOOLEAN DEFAULT FALSE,
    notification_type VARCHAR(20) NOT NULL
);
CREATE INDEX idx_notification_created_at ON Notification
(created_at DESC);

-- FormCorrection (notifications)
```

```
CREATE TABLE FormCorrection (
    id SERIAL PRIMARY KEY,
    analysis_id INTEGER REFERENCES ExerciseAnalysis(id) ON
DELETE CASCADE,
    timestamp FLOAT NOT NULL,
    segment VARCHAR(3) NOT NULL,
    severity SMALLINT NOT NULL,
    message TEXT NOT NULL,
    suggested_cue VARCHAR(200) NOT NULL
);
CREATE INDEX idx_formcorrection_timestamp ON FormCorrection
(timestamp);

-- WorkoutPlan (plans)
CREATE TABLE WorkoutPlan (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    creator_id INTEGER REFERENCES CustomUser(id) ON DELETE
CASCADE,
    is_public BOOLEAN DEFAULT FALSE,
    difficulty_level SMALLINT DEFAULT 1,
    duration_weeks SMALLINT DEFAULT 4,
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);

-- PlanSubscription (plans)
CREATE TABLE PlanSubscription (
    id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES CustomUser(id) ON DELETE
CASCADE,
    plan_id INTEGER REFERENCES WorkoutPlan(id) ON DELETE
CASCADE,
    start_date DATE NOT NULL,
    is_active BOOLEAN DEFAULT TRUE,
    current_week SMALLINT DEFAULT 1
);

-- PlannedWorkout (plans)
CREATE TABLE PlannedWorkout (
```

```
    id SERIAL PRIMARY KEY,
    plan_id INTEGER REFERENCES WorkoutPlan(id) ON DELETE
CASCADE,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    week_number SMALLINT NOT NULL,
    day_of_week SMALLINT NOT NULL,  -- 1=Monday, 7=Sunday
    estimated_duration INTERVAL
);
CREATE INDEX idx_plannedworkout_week_day ON PlannedWorkout
(week_number, day_of_week);

-- PlannedExercise (plans)
CREATE TABLE PlannedExercise (
    id SERIAL PRIMARY KEY,
    planned_workout_id INTEGER REFERENCES
PlannedWorkout(id) ON DELETE CASCADE,
    exercise_type_id INTEGER REFERENCES ExerciseType(id) ON
DELETE CASCADE,
    "order" SMALLINT DEFAULT 0,
    sets SMALLINT DEFAULT 3,
    reps VARCHAR(50) DEFAULT '8-12',
    rest_seconds SMALLINT DEFAULT 60,
    notes TEXT
);
CREATE INDEX idx_plannedexercise_order ON PlannedExercise
("order");

-- ResourceCategory (resources)
CREATE TABLE ResourceCategory (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    slug VARCHAR(100) UNIQUE NOT NULL,
    description TEXT
);

-- Article (resources)
CREATE TABLE Article (
    id SERIAL PRIMARY KEY,
    title VARCHAR(200) NOT NULL,
```

```
    slug VARCHAR(200) UNIQUE NOT NULL,
    author_id INTEGER REFERENCES CustomUser(id) ON DELETE
PROTECT,
    category_id INTEGER REFERENCES ResourceCategory(id) ON
DELETE PROTECT,
    content TEXT NOT NULL,
    summary TEXT,
    featured_image TEXT,
    published BOOLEAN DEFAULT FALSE,
    published_date TIMESTAMP,
    created_at TIMESTAMP NOT NULL DEFAULT
CURRENT_TIMESTAMP,
    updated_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);

-- Junction table for Article <-> ExerciseType (ManyToMany)
CREATE TABLE Article_ExerciseType (
    article_id INTEGER REFERENCES Article(id) ON DELETE
CASCADE,
    exercisetype_id INTEGER REFERENCES ExerciseType(id) ON
DELETE CASCADE,
    PRIMARY KEY (article_id, exercisetype_id)
);

-- Tutorial (resources)
CREATE TABLE Tutorial (
    id SERIAL PRIMARY KEY,
    title VARCHAR(200) NOT NULL,
    slug VARCHAR(200) UNIQUE NOT NULL,
    description TEXT NOT NULL,
    video_url TEXT NOT NULL,
    duration INTERVAL NOT NULL,
    difficulty_level SMALLINT DEFAULT 1,
    category_id INTEGER REFERENCES ResourceCategory(id) ON
DELETE PROTECT,
    published BOOLEAN DEFAULT FALSE,
    published_date TIMESTAMP,
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

```
-- Junction table for Tutorial <-> ExerciseType
(ManyToMany)
CREATE TABLE Tutorial_ExerciseType (
    tutorial_id INTEGER REFERENCES Tutorial(id) ON DELETE
CASCADE,
    exercisetype_id INTEGER REFERENCES ExerciseType(id) ON
DELETE CASCADE,
    PRIMARY KEY (tutorial_id, exercisetype_id)
);
```