REST API Documentation

==========================================================

1. AUTHENTICATION ENDPOINTS

----------------------------------------------------------

1.1 Create New Account

Endpoint: ~/api/account

Method: POST

Description: Creates a new user account with the provided email, name, and password.

Request Body:

{

"email": "string (required)",

"name": "string (required)",

"password": "string (required)"

}

Response:

• Success (200 OK): { "token": "string (user ID)" }

• Error (400 Bad Request): { "err": "missing not null value" }

----------------------------------------------------------

1.2 Login Account

Endpoint: ~/api/account

Method: GET

Description: Authenticates a user using email/username and password

Query Parameters:

{

"email": "string (optional)",

"name": "string (optional)",

"password": "string (required)"

}

Response:

• Success (200 OK): { "token": "string (user ID)" }

• Error (400 Bad Request): { "err": "missing not null value" }

• Error (401 Unauthorized): { "err": "password error" }

----------------------------------------------------------

1.3 Update Password

Endpoint: ~/api/account

Method: PUT

Description: Updates user's password. Requires authentication.

Request Body:

{

"currentPassword": "string (required)",

"newPassword": "string (required)"

}

Response:

• Success (200 OK): { "msg": "Password updated successfully" }

• Error (400 Bad Request): Error object

----------------------------------------------------------

1.4 Delete Account

Endpoint: ~/api/account

Method: DELETE

Description: Deletes user account. Requires authentication.

Request Body:

{

"password": "string (required)"

}

Response:

• Success (200 OK): { "msg": "Account deleted successfully" }

• Error (401 Unauthorized): { "err": "Invalid password" }

==========================================================

2. EXERCISE ENDPOINTS

----------------------------------------------------------

2.1 Create Exercise

Endpoint: ~/api/exercise

Method: POST

Description: Creates a new exercise. Requires authentication.

Request Body:

{

"name": "string (required)",

"type": "string (required)",

"description": "string (optional)",

"equipment": "string (optional)",

"reps": "number (optional)",

"sets": "number (optional)",

"duration": "number (optional)"

}

Response:

• Success (200 OK):

{

"msg": "Exercise created successfully",

"data": result.rows

}

• Error (400 Bad Request): { "err": "Missing required fields" }

----------------------------------------------------------

2.2 Get Exercise Types

Endpoint: ~/api/exercise/types

Method: GET

Description: Retrieves all exercise types. Public route.

Response:

• Success (200 OK): Array of exercise types

• Error (400 Bad Request): { "err": error\_message }

----------------------------------------------------------

2.3 Get Equipment List

Endpoint: ~/api/exercise/equipment

Method: GET

Description: Retrieves all available equipment. Public route.

Response:

• Success (200 OK): Array of equipment

• Error (400 Bad Request): { "err": error\_message }

----------------------------------------------------------

2.4 Get All Exercises

Endpoint: ~/api/exercise

Method: GET

Description: Retrieves all exercises (system + user's custom). Requires authentication.

Response:

• Success (200 OK): Array of exercises

• Error (400 Bad Request): { "err": error\_message }

----------------------------------------------------------

2.5 Search Exercises

Endpoint: ~/api/exercise/search

Method: GET

Description: Searches exercises based on criteria. Requires authentication.

Query Parameters:

• q: Search term (optional)

• type: Exercise type (optional)

• equipment: Equipment type (optional)

Response:

• Success (200 OK): Array of matching exercises

• Error (400 Bad Request): { "err": error\_message }

----------------------------------------------------------

2.6 Get Exercises by Type

Endpoint: ~/api/exercise/type/:type

Method: GET

Description: Retrieves exercises of specific type. Requires authentication.

Path Parameters:

• type: Exercise type

Response:

• Success (200 OK): Array of exercises

• Error (400 Bad Request): { "err": error\_message }

----------------------------------------------------------

2.7 Get User's Custom Exercises

Endpoint: ~/api/exercise/custom

Method: GET

Description: Retrieves user's custom exercises. Requires authentication.

Response:

• Success (200 OK): Array of custom exercises

• Error (400 Bad Request): { "err": error\_message }

----------------------------------------------------------

2.8 Get Specific Exercise

Endpoint: ~/api/exercise/:id

Method: GET

Description: Retrieves specific exercise by ID. Requires authentication.

Path Parameters:

• id: Exercise ID

Response:

• Success (200 OK): Exercise object

• Error (404 Not Found): { "err": "Exercise not found" }

----------------------------------------------------------

2.9 Update Exercise

Endpoint: ~/api/exercise/:id

Method: PUT

Description: Updates existing exercise. Requires authentication.

Path Parameters:

• id: Exercise ID

Request Body: Same as Create Exercise

Response:

• Success (200 OK): { "msg": "Exercise updated successfully" }

• Error (403 Forbidden): { "err": "Cannot modify system exercises" }

• Error (404 Not Found): { "err": "Exercise not found" }

----------------------------------------------------------

2.10 Delete Exercise

Endpoint: ~/api/exercise/:id

Method: DELETE

Description: Deletes an exercise. Requires authentication.

Path Parameters:

• id: Exercise ID

Response:

• Success (200 OK): { "msg": "Exercise deleted successfully" }

• Error (403 Forbidden): { "err": "Cannot delete system exercises" }

• Error (404 Not Found): { "err": "Exercise not found" }

----------------------------------------------------------

2.11 Clone Exercise

Endpoint: ~/api/exercise/:id/clone

Method: POST

Description: Creates a copy of existing exercise. Requires authentication.

Path Parameters:

• id: Exercise ID to clone

Response:

• Success (200 OK):

{

"msg": "Exercise cloned successfully",

"data": result.rows

}

• Error (404 Not Found): { "err": "Exercise not found" }

==========================================================

3. WORKOUT PLAN ENDPOINTS

----------------------------------------------------------

3.1 Create Plan

Endpoint: ~/api/plan

Method: POST

Description: Creates new workout plan. Requires authentication.

Request Body:

{

"exercise\_id": "number (required)",

"name": "string (required)",

"start\_date": "string (required)",

"frequency\_type": "string (required)",

"frequency\_value": "number (required)",

"description": "string (optional)",

"end\_date": "string (optional)",

"target\_sets": "number (optional)",

"target\_reps": "number (optional)",

"target\_duration": "number (optional)",

"days\_of\_week": "string (optional)",

"preferred\_time": "string (optional)",

"priority": "number (optional)"

}

Response:

• Success (200 OK): { "msg": "Plan created successfully", "data": result.rows }

• Error (400 Bad Request): { "err": "Missing required field: [field]" }

----------------------------------------------------------

3.2 Get All User Plans

Endpoint: ~/api/plan

Method: GET

Description: Retrieves all user's workout plans. Requires authentication.

Response:

• Success (200 OK): Array of workout plans

• Error (400 Bad Request): { "err": error\_message }

----------------------------------------------------------

3.3 Get Active Plans

Endpoint: ~/api/plan/active

Method: GET

Description: Retrieves user's active workout plans. Requires authentication.

Query Parameters:

• date: Target date (optional, defaults to current date)

Response:

• Success (200 OK): Array of active plans

• Error (400 Bad Request): { "err": error\_message }

----------------------------------------------------------

3.4 Get Plans by Frequency

Endpoint: ~/api/plan/frequency/:frequency\_type

Method: GET

Description: Retrieves plans by frequency type. Requires authentication.

Path Parameters:

• frequency\_type: "daily", "weekly", "monthly", or "custom"

Response:

• Success (200 OK): Array of matching plans

• Error (400 Bad Request): { "err": "Invalid frequency type" }

----------------------------------------------------------

3.5 Get Specific Plan

Endpoint: ~/api/plan/:id

Method: GET

Description: Retrieves specific workout plan. Requires authentication.

Path Parameters:

• id: Plan ID

Response:

• Success (200 OK): Plan object

• Error (403 Forbidden): { "err": "Access denied" }

• Error (404 Not Found): { "err": "Plan not found" }

----------------------------------------------------------

3.6 Update Plan

Endpoint: ~/api/plan/:id

Method: PUT

Description: Updates workout plan. Requires authentication.

Path Parameters:

• id: Plan ID

Request Body: Same as Create Plan (all fields optional)

Response:

• Success (200 OK): { "msg": "Plan updated successfully" }

• Error (404 Not Found): { "err": "Plan not found" }

----------------------------------------------------------

3.7 Update Plan Status

Endpoint: ~/api/plan/:id/status

Method: PATCH

Description: Updates plan status. Requires authentication.

Path Parameters:

• id: Plan ID

Request Body:

{

"status": "string (required)"

}

Response:

• Success (200 OK): { "msg": "Plan status updated successfully" }

• Error (400 Bad Request): { "err": "Status is required" }

----------------------------------------------------------

3.8 Delete Plan

Endpoint: ~/api/plan/:id

Method: DELETE

Description: Deletes workout plan. Requires authentication.

Path Parameters:

• id: Plan ID

Response:

• Success (200 OK): { "msg": "Plan deleted successfully" }

• Error (404 Not Found): { "err": "Plan not found" }

==========================================================

4. USER PROFILE ENDPOINTS

----------------------------------------------------------

4.1 Update Profile

Endpoint: ~/api/profile

Method: POST

Description: Updates user profile information

Request Body:

{

"token": "string (required)",

"first\_name": "string (optional)",

"last\_name": "string (optional)",

"avatar\_url": "string (optional)",

"introduction": "string (optional)",

"height": "number (optional)",

"weight": "number (optional)",

"fitness\_level": "number (optional)",

"fitness\_goals": "string (optional)",

"birthday": "timestamp (optional)",

"training\_start\_date": "timestamp (optional)",

"phone": "string (optional)",

"Email": "string (optional)",

"country": "string (optional)",

"city": "string (optional)",

"state": "string (optional)"

}

Response:

• Success (200 OK): Updated profile data

• Error (400 Bad Request): { "err": "missing token" }

----------------------------------------------------------

4.2 Get Profile

Endpoint: ~/api/profile

Method: GET

Description: Retrieves user profile information

Query Parameters:

• token: User token (required)

Response:

• Success (200 OK): Profile data

• Error (400 Bad Request): { "err": "missing token" }

----------------------------------------------------------

4.3 Update Avatar

Endpoint: ~/api/profile/avatar

Method: POST

Description: Updates user's profile picture

Request:

• Multipart form data with 'avatar' field

Response:

• Success (200 OK): { "success": "updateAvatar" }

• Error (400 Bad Request): Error object

----------------------------------------------------------

4.4 Get Avatar

Endpoint: ~/api/profile/avatar

Method: GET

Description: Retrieves user's profile picture

Query Parameters:

• token: User token (required)

Response:

• Success (200 OK): Image file

• Error (404 Not Found): { "err": "no such file" }

==========================================================

End of Documentation

2.1 Get Exercise Types (updated route): ~/api/exercise/typelist (GET)

2.2 Get Equipment List (updated route): ~/api/exercise/equipmentlist (GET)

2.3 Get Exercises by Type (added): ~/api/exercise/type (GET)

2.4 Get Exercises by Equipment (added): ~/api/exercise/equipment (GET)

2.5 Get Exercises by ID (updated route): ~/api/exercise/id (GET)

2.6 Delete Exercise (updated): ~/api/exercise (DELETE)

Note: All routes require authentication under ~/api/plan

## 5. USER RECORD ENDPOINTS

5.1 Create Record: ~/api/record (POST)

5.2 Delete Record: ~/api/record (DELETE)

5.3 Update Record: ~/api/record (PUT)

5.4 Search Record: ~/api/record (GET)