**StrengthStack MVP: Service Endpoints Design**

**1. Service Endpoints (URLs)**

The following service endpoints will be implemented for the StrengthStack MVP:

* /api/users/register (POST)
* /api/users/login (POST)
* /api/users/{userId} (GET)
* /api/users/{userId} (PUT)
* /api/workouts (POST)
* /api/workouts/{workoutId} (GET)
* /api/workouts/user/{userId} (GET)
* /api/exercises (GET)
* /api/exercises/{exerciseId} (GET)
* /api/progress/user/{userId} (GET)
* /api/progress/workout/{workoutId} (GET)
* /api/progress/graph/{userId} (GET)
* /api/recommendations/{userId} (GET)
* /api/calendar/{userId} (GET)

**2. Endpoint Purposes and User Persona Goals**

* **/api/users/register (POST):**
  + **Purpose:** Creates a new user account.
  + **Persona Goal:** Allows Beginner Ben and Progressive Patty to create an account and store their profile data (username, password, training level, goals).
  + **Data Change:** Adds a new record to the Users table.
* **/api/users/login (POST):**
  + **Purpose:** Authenticates a user and returns a session token.
  + **Persona Goal:** Allows Ben and Patty to access their account and personalized data.
  + **Data Change:** No direct data change, but establishes a session.
* **/api/users/{userId} (GET):**
  + **Purpose:** Retrieves a specific user's profile information.
  + **Persona Goal:** Allows users to view their profile details.
  + **Data Change:** No data change.
* **/api/users/{userId} (PUT):**
  + **Purpose:** Updates a specific user's profile information.
  + **Persona Goal:** Allows users to modify their profile data (e.g., training level, goals).
  + **Data Change:** Updates a record in the Users table.
* **/api/workouts (POST):**
  + **Purpose:** Logs a new workout.
  + **Persona Goal:** Allows Ben and Patty to record their workout data (sets, reps, weights).
  + **Data Change:** Adds a new record to the Workouts table.
* **/api/workouts/{workoutId} (GET):**
  + **Purpose:** Retrieves a specific workout's details.
  + **Persona Goal:** Allows users to view the details of a logged workout.
  + **Data Change:** No data change.
* **/api/workouts/user/{userId} (GET):**
  + **Purpose:** Retrieves all workouts for a specific user.
  + **Persona Goal:** Allows users to view their workout history.
  + **Data Change:** No data change.
* **/api/exercises (GET):**
  + **Purpose:** Retrieves a list of all exercises.
  + **Persona Goal:** Allows users to view available exercises for workout planning.
  + **Data Change:** No data change.
* **/api/exercises/{exerciseId} (GET):**
  + **Purpose:** Retrieves a specific exercise's details.
  + **Persona Goal:** Allows users to view the details of a specific exercise.
  + **Data Change:** No data change.
* **/api/progress/user/{userId} (GET):**
  + **Purpose:** Retrieves all progress records for a specific user.
  + **Persona Goal:** Allows users to view their overall progress.
  + **Data Change:** No data change.
* **/api/progress/workout/{workoutId} (GET):**
  + **Purpose:** Retrieves all progress records for a specific workout.
  + **Persona Goal:** Allows users to view the progress of a specific workout.
  + **Data Change:** No data change.
* **/api/progress/graph/{userId} (GET):**
  + **Purpose:** Retrieves progress data for generating graphs.
  + **Persona Goal:** Allows users to visualize their progress over time.
  + **Data Change:** No data change.
* **/api/recommendations/{userId} (GET):**
  + **Purpose:** Retrieves personalized workout recommendations.
  + **Persona Goal:** Allows users to receive personalized workout suggestions.
  + **Data Change:** No data change.
* **/api/calendar/{userId} (GET):**
  + **Purpose:** Retrieves workout data for the calendar view.
  + **Persona Goal:** Allows users to view their workout schedule.
  + **Data Change:** No data change.

**3. Example Requests and Responses (JSON)**

* **/api/users/register (POST):**
  + **Request (Success):**

JSON

{

"username": "patty.rodriguez",

"password": "securePass123",

"trainingLevel": "advanced",

"goals": "strength"

}

* + **Response (Success):**

JSON

{

"message": "User registered successfully",

"userId": 2

}

* + **Response (Error - Duplicate Username):**

JSON

{

"error": "Username already exists",

"statusCode": 409

}

* + **Response (Error - Invalid Training Level):**

JSON

{

"error": "Invalid trainingLevel. Must be 'beginner', 'intermediate', or 'advanced'.",

"statusCode": 400

}

* + **Response (Error - Missing Required Field):**

JSON

{

"error": "Missing required field: password",

"statusCode": 400

}

* **/api/users/login (POST):**
  + **Request (Success):**

JSON

{

"username": "patty.rodriguez",

"password": "securePass123"

}

* + **Response (Success):**

JSON

{

"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."

}

* + **Response (Error - Invalid Credentials):**

JSON

{

"error": "Invalid username or password",

"statusCode": 401

}

* + **Response (Error - Account Locked):**

JSON

{

"error": "Account is locked due to too many failed login attempts.",

"statusCode": 403

}

* **/api/workouts (POST):**
  + **Request (Success):**

JSON

{

"userId": 2,

"exerciseId": 3,

"date": "2024-10-28",

"sets": 4,

"reps": 8,

"weight": 100,

"rir": 1,

"rpe": 9

}

* + **Response (Success):**

JSON

{

"message": "Workout logged successfully",

"workoutId": 2

}

* + **Response (Error - Invalid UserID):**

JSON

{

"error": "Invalid User ID",

"statusCode": 404

}

* + **Response (Error - Invalid ExerciseID):**

JSON

{

"error": "Invalid Exercise ID",

"statusCode": 404

}

* + **Response (Error - Invalid Reps/Weight):**

JSON

{

"error": "Invalid input. Reps and weight must be positive numbers.",

"statusCode": 400

}

* /api/progress/graph/{userId} **(GET):**
  + **Response (Success):**

JSON

[

{

"date": "2024-10-21",

"weight": 100,

"reps": 8,

"estimatedOneRepMax": 120

},

{

"date": "2024-10-28",

"weight": 105,

"reps": 9,

"estimatedOneRepMax": 125

}

]

* + **Response (Error - User Not Found):**

JSON

{

"error": "User not found",

"statusCode": 404

}

* + **Response (Error - No Progress Data):**

JSON

{

"error": "No progress data available for this user.",

"statusCode": 204

}

* /api/calendar/{userId} **(GET):**
  + **Response (Success):**

JSON

[

{

"date": "2024-10-28",

"workoutId": 2,

"exercises": ["Bench Press","Squats"]

},

{

"date": "2024-10-30",

"workoutId": 3,

"exercises": ["Deadlifts", "Rows"]

}

]

* + **Response (Error - User Not Found):**

JSON

{

"error": "User not found",

"statusCode": 404

}

**4. Communication Diagram**

sequenceDiagram

participant User Interface (UI)

participant API Service

participant Database

API Service

Database

UI

UI->>API Service: /api/users/register (POST)

API Service->>Database: Insert new user record

Database-->>API Service: Success/Failure

API Service-->>UI: Response (Success/Error)

UI->>API Service: /api/users/login (POST)

API Service->>Database: Query user credentials

Database-->>API Service: User data

API Service-->>UI: Response (Token/Error)

UI->>API Service: /api/workouts (POST)

API Service->>Database: Insert new workout record

Database-->>API Service: Success/Failure

API Service-->>UI: Response (Success/Error)

UI->>API Service: /api/progress/graph/{userId} (GET)

API Service->>Database: Query progress data

Database-->>API Service: Progress data

API Service-->>UI: Response (Progress data)

UI->>API Service: /api/calendar/{userId} (GET)

API Service->>Database: Query workout data

Database-->>API Service: workout data

API Service-->>UI: Response (calendar data)

**1. Are you using RESTful principles when they are appropriate?**

* **Yes.** The API demonstrates a clear adherence to RESTful principles:
  + Resources are identified by URLs.
  + Standard HTTP methods are used appropriately.
  + The API is designed to be stateless.
  + JSON is used for data exchange.
* These choices are appropriate for a web application that manages structured data and requires clear separation between the client and server.

**2. Are HTTP verbs used appropriately and in a consistent way?**

* **Yes.**
  + POST is consistently used for creating new resources (users, workouts).
  + GET is consistently used for retrieving resources (users, workouts, exercises, progress data).
  + PUT is used for updating resources (user data).
* This consistent usage aligns with the semantic meaning of HTTP verbs.

**3. Do paths logically correspond to their purpose and effectively communicate that purpose?**

* **Yes.**
  + Paths like /api/users, /api/workouts, and /api/exercises clearly indicate the resources being accessed.
  + The use of {userId} and {workoutId} in paths effectively communicates that specific resources are being targeted.
  + /api/progress/graph/{userId} clearly shows that it is used to retrieve progress graph data.
  + /api/calendar/{userId} clearly shows that it is used to retrieve calendar data.
* The paths are well-structured and easy to understand.

**4. Can I or another experienced software engineer/developer build your application using the mockups from last week, your ERD due this week AND this design document? Is it complete and show ALL of the necessary calls from each of your pages to the backend?**

* **Mostly, with some considerations.**
  + The design document provides a comprehensive overview of the API endpoints, their purposes, and example requests/responses.
  + The ERD provides the necessary information about the database structure.
  + The Communication diagram, and page to endpoint mapping, provides a good overview of what pages will call what endpoints.

**In summary:** The StrengthStack API design demonstrates a strong adherence to RESTful principles, uses HTTP verbs appropriately, and provides clear and logical paths. With the provided documentation, an experienced developer should be able to build the application's backend.