

Service Layer

**Cristian Ferreira**

**July 16, 2023**

**FitAll**

The FitAll Capstone Project aims to develop a web application that promotes fitness and provides users with the tools to track their progress and achieve their fitness goals. This service design document outlines the architecture and endpoints required to create a seamless user experience and ensure efficient data management. By following RESTful principles and utilizing appropriate HTTP verbs, we can establish a scalable and extensible software architecture that conforms to industry best practices.

Service endpoints have been identified and categorized into User Management, Workout Management, and Progress Tracking. Each endpoint serves a specific purpose in relation to user data and is designed to handle requests and provide appropriate responses. Example requests and responses are provided, including error handling scenarios to ensure a smooth user experience.

Additionally, communication diagrams illustrate the flow of information from the user interface pages to the service endpoints, ensuring that the designed services and endpoints fulfill the requirements of the Minimum Viable Product (MVP) without including unnecessary functionalities. This document will serve as a comprehensive guide for developers to build the FitAll web application, utilizing the provided mockups and Entity-Relationship Diagram (ERD).

**Overview**

FitAll requires a service layer to facilitate the interaction between the user interface and the database. The service layer will handle the necessary operations and communication with the backend to ensure seamless functionality. This document outlines the service endpoints, their purposes, example requests and responses, error handling, and communication diagrams to provide a comprehensive overview of the service design.

**Service Endpoints**

**User Management**

Endpoint: /api/users/register

* Purpose: Register a new user in the system.
* Method: POST
* Request Body:

{

"username": "cristianferreira",

"email": "cristianferreira@mail.com",

"password": "Example"

}

Response

{

"message": "User registered successfully",

"user": {

"id": "123456789",

"username": " cristianferreira ",

"email": " cristianferreira@mail.com "

}

}

Endpoint: /api/users/login

* Purpose: Authenticate and login an existing user.
* Method: POST
* Request Body

{

"username": "" cristianferreira ",

"password": " Example "

}

Response

{

"message": "Login successful",

"user": {

"id": "123456789",

"username": " cristianferreira ",

"email": " cristianferreira@mail.com "

},

"token": "hdhfmfajrjiscpo”

}

Endpoint: /api/users/profile

* Purpose: Fetch the user's profile information.
* Method: GET
* Headers:

{

"id": "123456789",

"username": " cristianferreira ",

"email": " cristianferreira@mail.com ",

"fullName": "Cristian Ferreira",

"age": 30,

"gender": "Male",

"address": "123 Main St",

"phone": "555-123-1313"

}

**Workout Management**

Endpoint: /api/workouts

* Purpose: Create a new workout.
* Method: POST
* Request Body:

{

"title": "Full Body Workout",

"description": "A comprehensive workout targeting all major muscle groups",

"exercises": [

{

"name": "Push-ups",

"sets": 3,

"reps": 10

},

{

"name": "Squats",

"sets": 3,

"reps": 12

}

]

}

Response

{

"message": "Workout created successfully",

"workout": {

"id": "987654321",

"title": "Full Body Workout",

"description": "A comprehensive workout targeting all major muscle groups",

"exercises": [

{

"name": "Push-ups",

"sets": 3,

"reps": 10

},

{

"name": "Squats",

"sets": 3,

"reps": 12

}

]

}

}

Endpoint: /api/workouts/:id

* Purpose: Get a specific workout by ID.
* Method: GET
* Response:

{

"id": "987654321",

"title": "Full Body Workout",

"description": "A comprehensive workout targeting all major muscle groups",

"exercises": [

{

"name": "Push-ups",

"sets": 3,

"reps": 10

},

{

"name": "Squats",

"sets": 3,

"reps": 12

}

]

}

**Progress Tracking**

Endpoint: /api/progress/weight

* Purpose: Record a weight measurement for the user.
* Method: POST
* Request Body:

{

"userId": "123456789",

"weight": 75.5,

"date": "2023-07-16"

}

Response

{

"message": "Weight recorded successfully",

"measurement": {

"id": "987654321",

"userId": "123456789",

"weight": 75.5,

"date": "2023-07-16"

}

}

Endpoint: /api/progress/user/:userId

* Purpose: Get the progress data for a specific user.
* Method: GET
* Response:

{

"userId": "123456789",

"weightMeasurements": [

{

"id": "987654321",

"weight": 75.5,

"date": "2023-07-16"

}

],

"bodyMeasurements": [

{

"id": "123456789",

"chest": 40,

"waist": 30,

"hips": 40,

"date": "2023-07-16"

}

],

"fitnessGoals": [

{

"id": "987654321",

"description": "Lose 10 pounds",

"targetDate": "2023-09-30"

}

]

}

**Error Handling**

The service endpoints handle error scenarios and provide meaningful error messages to the user. Here are some examples of erroneous requests and their corresponding error responses:

Invalid User Registration Request:

* Request: POST /api/users/register
* Request Body:

{

"email": "cristian@mail.com",

"password": "example"

}

Response

{

"error": "Invalid registration request",

"message": "Username is required"

}

Unauthorized Access:

* Request: `GET /api/work

The appropriateness of the endpoint design will be assessed based on adherence to RESTful principles, consistent use of HTTP verbs, and logical correspondence between endpoint paths and their intended purposes. The document will also be evaluated for completeness, ensuring that all necessary calls from each user interface page to the backend are included. Sample requests and responses, along with error handling, will be reviewed to confirm the design's practicality and adherence to industry standards.

**Conclusion**

In conclusion, the FitAll Capstone Project service design document presents a comprehensive overview of the service layer architecture, endpoints, and communication flow required to develop the FitAll web application. By following RESTful principles and utilizing appropriate HTTP verbs, the document ensures scalability and extensibility while maintaining a user-centric approach to data management. The provided sample requests and responses, along with error handling scenarios, demonstrate the robustness of the designed services