All the APIs will be accessed with respect to localhost as the base url. And the server and client are communicating via the HTTP protocol.

We have three microservices (MealService, WorkoutService, UserService) and Service Registry and they use HTTP protocol to communicate with client (web client and desktop client) and other microservices.

We have two types of client – web client and a desktop client, and these will interact with the Service Registry to route the request to the target microservice. Service Registry is another service in itself, which uses a concurrent hash map to maintain the connection details (Port number, etc) of all the microservices which are up and running. If any microservice is down, the respective service will be deregistered in the Service registry ( and the related record in the hashmap is removed).

For example, the below validateUser API (mentioned below) will be accessed as :

http://localhost:8083[/userService/api/user/validateUser?email=sid@gmail.com&password=password](mailto:/userService/api/user/validateUser?email=sid@gmail.com&password=password)

Let’s go through the list of APIs exposed by each service in our system.

**APIs of UserService**

**LOGIN**

REQUEST:

(POST) http://localhost:8083[/userService/api/user/validateUser?email=sid@gmail.com&password=password](mailto:/userService/api/user/validateUser?email=sid@gmail.com&password=password) HTTP/1.1

RESPONSE OUTPUT:

(Content-type: “application/json;charset=UTF-8”)

{

"admin": 0,

"emailId": "sid@gmail.com",

"name": "Siddharrth",

"password": "password",

"phoneNumber": "8324567869",

"userID": 1

}

Response code: 200 (if successful)

Response code: 406 or 500 (if failure)

**USER REGISTRATION**

REQUEST:

(POST) <http://localhost:8083/userService/api/user/registerUser?email=test@gmail.com&password=test&phone=8768738271&name=testuser> HTTP/1.1

Response output:

(Content-type: “text/html”)

"ACCEPTED"

Response code: 200 OK

Response code: 500 (if failure)

**APIs of WorkoutService**

**GET WORKOUT PLAN OF A PARTICULAR USER (By USER ID)**

REQUEST LINE:

(GET) <http://localhost:8081/workoutService/api/workout/getWorkoutPlan?id=1> HTTP/1.1

RESPONSE OUTPUT:

(Content-type: “application/json;charset=UTF-8”)

[

{

"description": "perpendicular posture recommended",

"name": "push-ups",

"targetUnitsPerDay": 10,

"workoutId": 1,

"workoutPlanId": 1,

"workoutPlanName": "Cardio workout"

},

{

"description": "high weight not recommended",

"name": "dead lift",

"targetUnitsPerDay": 20,

"workoutId": 3,

"workoutPlanId": 1,

"workoutPlanName": "Cardio workout"

}

]

RESPONSE CODE: 200 OK (if successful)

For sad path, (Where UserID is not valid)

Response code : 500

**GET STATUS OF ALL AVAILABLE WORKOUTS WITH RESPECT TO A PARTICULAR USER:**

REQUEST LINE:

(GET) <http://localhost:8081/workoutService/api/workout/getWorkoutsStatusForUser?id=1> HTTP/1.1

RESPONSE OUTPUT:

(Content-type: “application/json;charset=UTF-8”)

[

{

"description": "perpendicular posture recommended",

"name": "push-ups",

"targetUnitsPerDay": 10,

"workoutId": 1,

"workoutPlanId": 1

},

{

"description": "back must be on the floor",

"name": "sit-ups",

"targetUnitsPerDay": 0,

"workoutId": 2,

"workoutPlanId": -1

},

{

"description": "high weight not recommended",

"name": "dead lift",

"targetUnitsPerDay": 20,

"workoutId": 3,

"workoutPlanId": 1

},

{

"description": "Do with the hands stretched forward",

"name": "squats",

"targetUnitsPerDay": 0,

"workoutId": 4,

"workoutPlanId": -1

}

]

For sad path, (Where UserID is not valid)

Response code : 500

**GET WORKOUT PROGRESS OF A PARTICULAR USER (GIVEN BY USER ID)**

REQUEST LINE:

(GET) <http://localhost:8081/workoutService/api/workout/getWorkoutProgress?id=1> HTTP/1.1

RESPONSE OUTPUT:

(Content-type: “application/json;charset=UTF-8”)

[

{

"dateOfWorkout": "01-10-2023",

"name": "dead lift",

"noOfUnitsPerformed": 10,

"workoutId": 3,

"workoutPlanId": 1

},

{

"dateOfWorkout": "02-10-2023",

"name": "push-ups",

"noOfUnitsPerformed": 8,

"workoutId": 1,

"workoutPlanId": 1

}

]

RESPONSE CODE: 200 OK

For sad path, (Where UserID is not valid)

Response code : 500

**UPDATE WORKOUT PLAN OF A USER:**

REQUEST LINE:

(POST) <http://localhost:8081/workoutService/api/workout/updateWorkoutPlan> HTTP/1.1

Request body:

(Content-type: “application/json;charset=UTF-8”)

[

{

"WorkoutId": 2,

"UserId": 1,

"WorkoutPlanId": 1,

"Quantity": 12

},

{

"WorkoutId": 1,

"UserId": 1,

"WorkoutPlanId": 1,

"Quantity": 0

},

{

"WorkoutId": 3,

"UserId": 1,

"WorkoutPlanId": 1,

"Quantity": 0

},

{

"WorkoutId": 4,

"UserId": 1,

"WorkoutPlanId": 1,

"Quantity": 0

}

]

Response code : 200 OK (FOR SUCCESS)

500 (FOR FAILURE)

**UPDATE WORKOUT PROGRESS OF A USER:**

REQUEST LINE:

(POST) <http://localhost:8081/workoutService/api/workout/updateWorkoutProgress> HTTP/1.1

Request body:

(Content-type: “application/json;charset=UTF-8”)

[

{

"WorkoutId": 2,

"UserId": 1,

"WorkoutPlanId": 1,

"Quantity": 32,

"DateOfWorkout": "11-10-2023"

},

{

"WorkoutId": 1,

"UserId": 1,

"WorkoutPlanId": 1,

"Quantity": 61,

"DateOfWorkout": "11-10-2023"

}

]

Response code : 200 OK (FOR SUCCESS)

500 (FOR FAILURE)

**UPDATE WORKOUT**

REQUEST LINE:

(POST) <http://localhost:8081/workoutService/api/workout/updateWorkout> HTTP/1.1

Request body:

(Content-type: “application/json;charset=UTF-8”)

[

{

"Name": "Sprinting",

"CaloriesBurnt": 10.50,

"Description": "100m fast running"

}

]

Response code : 200 OK (FOR SUCCESS)

500 (FOR FAILURE)

**APIs of MealService**

**GET MEAL PLAN**

REQUEST LINE:

(GET) <http://localhost:8082/mealService/api/meal/getMealPlan?id=1> HTTP/1.1

RESPONSE:

(Content-type: “application/json;charset=UTF-8”)

[

{

"userID": 1,

"mealPlanID": 1,

"mealID": 1,

"targetUnitsPerDay": 10,

"targetCalorieCount": 100.0,

"mealType": "Lunch"

}

]

Response code : 200 OK (FOR SUCCESS)

500 (FOR FAILURE)

**GET THE STATUS OF ALL MEALS (FOOD ITEMS) WRT THE MEAL PLAN OF THE GIVEN USER (identified by the user ID passed as query parameter)**

REQUEST LINE:

(GET) <http://localhost:8082/mealService/api/meal/getMealStatusForUser?id=1> HTTP/1.1

RESPONSE:

(Content-type: “application/json;charset=UTF-8”)

[

{

"mealName": "Egg",

"mealId": 1,

"mealPlanId": 1,

"targetUnitsPerDay": 10,

"description": "Without yellow yolk",

"mealType": "Lunch"

},

{

"mealName": "Chicken",

"mealId": 2,

"mealPlanId": -1,

"targetUnitsPerDay": 0,

"description": "Grilled or baked",

"mealType": "None"

}

]

Response code : 200 OK (FOR SUCCESS)

500 (FOR FAILURE)

**GET THE MEAL INTAKE RECORDS OF THE GIVEN USER (identified by the user ID passed as query parameter)**

REQUEST LINE:

(GET) <http://localhost:8082/mealService/api/meal/getMealIntake?id=1>

HTTP/1.1

RESPONSE:

(Content-type: “application/json;charset=UTF-8”)

[

{

"userID": 1,

"mealPlanID": 1,

"mealID": 1,

"unitsDone": 5,

"dateOfMeal": "30-09-2023",

"mealType": "Lunch"

}

]

Response code : 200 OK (FOR SUCCESS)

500 (FOR FAILURE)

**UPDATE MEAL PLAN OF USER :**

REQUEST LINE:

(POST) <http://localhost:8082/mealService/api/meal/updateMealPlan> HTTP/1.1

Request body:

(Content-type: “application/json;charset=UTF-8”)

[

{

"userId": 1,

"mealPlanId": 1,

"mealId": 1,

"quantity": 4,

"mealType": "Breakfast"

},

{

"userId": 1,

"mealPlanId": 1,

"mealId": 2,

"quantity": 3,

"mealType": "Dinner"

}

]

Response code : 200 OK (FOR SUCCESS)

500 (FOR FAILURE)

**UPDATE THE MEAL INTAKE FOR A USER:**

REQUEST LINE:

(POST) <http://localhost:8082/mealService/api/meal/updateMealIntake> HTTP/1.1

Request body:

(Content-type: “application/json;charset=UTF-8”)

[

{

"userID": 1,

"mealPlanID": 1,

"mealID": 1,

"unitsDone": 5,

"dateOfMeal": "10-10-2023",

"mealType": "Dinner"

}

]

Response code : 200 OK (FOR SUCCESS)

500 (FOR FAILURE)

**UPDATE MEAL DETAILS:**

REQUEST LINE:

(POST) <http://localhost:8082/mealService/api/meal/updateMeal> HTTP/1.1

Request body:

(Content-type: “application/json;charset=UTF-8”)

[

{

"Name": "Caesar salad",

"Calories": 76.50,

"Description": "Without Mayo and Egg"

}

]

Response code : 200 OK (FOR SUCCESS)

500 (FOR FAILURE)

**APIs of ServiceRegistry**

**REGISTER SERVICE**

This API is used to register the service in the service registry.

Name attribute can have “meal”, “workout”, “user” (which indicates the set of microservices in our system)

REQUEST LINE:

(POST) <http://localhost:8080/serviceRegistry/api/service/registerService?name=meal&portNumber=8082> HTTP/1.1

Response code : 200 OK (FOR SUCCESS)

500 (FOR FAILURE)

**DEREGISTER SERVICE**

This API is used to deregister the service in the service registry.

Name attribute can have “meal”, “workout”, “user” (which indicates the set of microservices in our system)

REQUEST LINE:

(POST) <http://localhost:8080/serviceRegistry/api/service/registerService?name=meal> HTTP/1.1

Response code : 200 OK (FOR SUCCESS)

500 (FOR FAILURE)

**GET SERVICE ADDRESS**

This API is used to fetch the server details (port number etc.) of the given service (indicated by its name in the query parameter).

**(GET)** <http://localhost:8080/serviceRegistry/api/service/getServiceAddress?name=user> HTTP/1.1

RESPONSE :

{

"portNumber": 8083

}

Response code : 200 OK (FOR SUCCESS)

500 (FOR FAILURE)