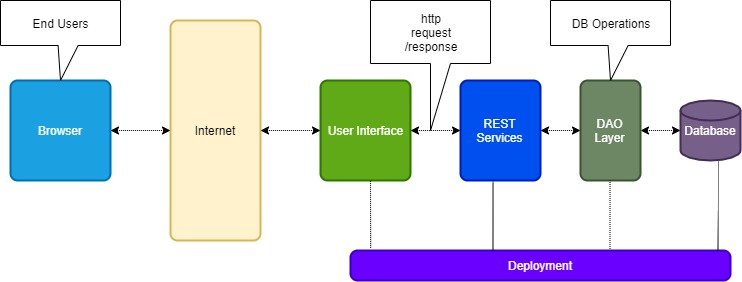
# **Fitness – Architecture of the Web Application**

The overall architecture of the web application is given below:



Browser resides on the client side and end users access the web application using browser through internet.

User interface is decoupled from the service layer. User interface can be built on any client side technologies such as HTML, JavaScript, jQuery, CSS or using any framework such as Angular, React etc.

Service layer or REST services expose different resources on HTTP protocol and it is decoupled from user interface or any layer. The REST services can be built using different technologies such as Java, Python, PHP etc. Here I am going to implement REST services using Python based Flask API.

DAO (Data Access Object) layer basically separated from the REST services so that this layer will be responsible only for handling database operations such as basic CRUD (Create, Read, Update, Delete) operations.

Database or persistence storage is used to store data permanently and the operations on such persistence storage happen through service layer. We can choose document (NoSQL) or relational (RDBMS). Here I am going to use MySQL database.

User Interface, REST services, DAO layer and persistence storage are deployed on their respective servers.

# **Fitness – Service Layer - REST Resources**

The below REST endpoints expose the services required for fitness web application.

**Testimonials**

This endpoint retrieves all testimonials which are written by people on this fitness portal from the database.

GET /testimonials

Response:

[

{

"name": "Chris Fox",

"profession": "CEO at Xcell Solutions",

"comment": "This website helped me with my fitness goals."

"profile\_imae": "/image\_chris.jpg",

“user\_id”: 1

},

{

"name": "Carolyn Kamper",

"profession": "Doctor",

"comment": "I recommend MFB to all my patients."

"profile\_imae": "/image\_carolyn.jpg",

“user\_id”: 2

}

]

This below endpoint retrieves a particular testimonial for a given testimonial id.

GET /testimonial/{testimonial\_id}

Response: see the above response for all testimonials but here only one testimonial will be returned.

The below endpoint will add new testimonial. A testimonial can be written by registered user or by any person who visits the portal.

POST /testimonial

Request Body:

{

"name": "Chris Fox",

"profession": "CEO at Xcell Solutions",

"comment": "This website helped me with my fitness goals."

"profile\_imae": "/image\_chris.jpg" ,

“user\_id”: 1

}

Response: 200 OK

**Subscription**

The below service endpoint retrieves all subscriptions details.

GET /subscriptions

Response:

[

{

"name": "Chris Fox",

"email": "chris@email.com",

"daily": true,

"active": true,

“user\_id”: 1

},

{

"name": "Carolyn Fox",

"email": "carolyn@email.com",

"daily": false,

"active": true,

“user\_id”: 1

}

]

The below service endpoint subscription details for a user.

GET /subscription/{user\_id}

For response please see the above response for all subscriptions but here only one subscription will be returned for a given user.

The below endpoint adds a new subscription when a person opts for a newsletter.

POST /subscription

Request Body:

{

"name": "Chris Fox",

"email": "chris@email.com",

"daily": true,

"active": true,

“user\_id”: 1

}

Response: 200 OK

The below endpoint deactivate exiting subscription when a person opts out for newsletter.

PUT /subscription

Request Body:

{

“id”: 1,

"active": "no

}

Response: 200 OK

**Contact Us**

When a person contact for a specific reason then he/she may contact us through contact us page. The below endpoint retrieves all issues from database.

GET /contacts

Response:

[

{

"name": "Chris Fox",

"email": "chris@email.com",

"country": "America"

"message": "This is a sample message",

“user\_id”: 2

},

{

"name": "Carolyn Fox",

"email": "carolyn@email.com",

"country": "Mexico"

"message": "This is a sample message",

“user\_id”: 1

}

]

The below endpoint retrieves single issue from the database. This returns only one issue details.

GET /contact/{contact\_id}

When a person contact for a specific reason then he/she may contact us through contact us page. This endpoint stores a new issue into database.

POST /contact

Request Body:

{

"name": "Chris Fox",

"email": "chris@email.com",

"country": "America"

"message": "This is a sample message",

“user\_id”: 1

}

Response: 200 OK

Once resolution is given to the issue a user may wish to close the ticket.

PUT /contact

{

“id”: 1,

“resolved”: “yes”

}

Response: 200 OK.

**Weight Loss Calculation**

This endpoint calculate weigh loss criteria and returns in a response.

POST /calculate/weightloss

Request Body:

{

"weight": 120,

"height": 72,

"age": 32,

"body\_type": "upper"

}

Response: {"calory" : 120}

**Muscle Gain Calculation**

This endpoint calculate muscle gain criteria and returns in a response.

POST /calculate/musclegain

Request Body:

{

"weight": 120,

"gain\_target": 72,

"age": 32

}

Response: {"protein" : 120}

**Maintain Weight Calculation**

This endpoint calculate maintain weight criteria and returns in a response.

POST /calculate/maintainweight

Request Body:

{

"weight": 120,

"height": 72,

"age": 32

}

Response: {"calory" : 120}

**Fitness Products**

Get all fitness products for displaying on the website.

GET /fitness/products

Response:

[

{

"name": "Shaker Bottle",

"price": "$5.99",

"image": "/shaker.jpg"

"buy\_url": "https://www.amazon.com/buy/q?id=1"

},

{

"name": "Exercise Ball",

"price": "$15.99",

"image": "/ball.jpg"

"buy\_url": "https://www.amazon.com/buy/q?id=2"

}

]

**Fitness Excercises**

Get all fitness exercises.

GET /fitness/execises

Response:

[

{

"body\_type": "upper",

"image": null,

"video\_url": "https://www.youtube.com/exercise/q?id=1"

},

{

"body\_type": "lower",

"image": null,

"video\_url": "https://www.youtube.com/exercise/q?id=1"

}

]

**User Registration**

User will store his/her personal details.

POST /fitness/user/registration

Request Body:

{

“name”: “full name”,

"email": "email@email.com",

"pwd": “encrypted password”,

"weight": 120,

“height”: 5.11,

“age”: 30,

“admin”: “no”

}

Response: 200 OK

**User Update**

User may want to update his/her personal details.

PUT /fitness/user/update

Request Body:

{

“id”: 1,

“name”: “full name”,

"email": "email@email.com",

"pwd": “encrypted password”,

"weight": 120,

“height”: 5.11,

“age”: 30

}

Response: 200 OK

**User Login**

Allow user login to his account on this portal using correct credentials.

POST /fitness/user/login

Request:

{

"email": "email@email.com",

"pwd": “plain text password”

}

Response:

{

“name”: “name”

}

**User Products**

Get all products a user bought.

GET /fitness/user/{user\_id}/products

[

{

<all product details>

}

]

**User Exercises**

Retrieves exercise details for a user.

GET /fitness/user/{user\_id}/exercises

[

{

<all exercise details>

}

]

**Delete User Exercise**

A user may not want to keep a exercise into his list.

DELETE /fitness/user/{user\_id}/exercise/{exercise\_id}

Response: 200 OK

**Error handling**

For REST service endpoints the below error will be handled.

When user sends bad input then http status code 404 is returned with below format:

{

“error\_code”: 404,

“error\_message”: “Bad user input request”

}

The below response is returned while there is server related errors – http status code is 5xx.

{

“error\_code”: 5xx,

“error\_message”: “Internal Server error”

}

There might be more REST services required depending on the actual implementation of the web application.