**User Registration App**

User registration API helps you register users in database in a secure way . 🚀

**Users**

You will be able to:

* **Create users**
* **Get users from DB**
* **Update users' credentials**
* **Delete users**

### Welcome to User registration API docs !

**GET**

**GET Request to get all users' data from database**

URL: <http://127.0.0.1:8000/users>

**This is to get details of all the users available in database**

This endpoint will let you have exposure to all the users signed up in database . just provide skip and limit values as query parameters to filter out number of records that you want to see. and API GET request will handle the rest for you. but make sure you authenticate yourself first in API otherwise you won't be authorized to use it

**Parameters:**

{Query Param} Skip : Integer ( ID of first record )

{Query Param} Limit : Integer ( ID of Last record )

Response:

Code: Description

200 Successful Response

Example Schema: [

{

"name": "Joelay field",

"dob": "2000-12-11",

"email": "joe@xyz.com",

"password": "any"

}

]

422 Validation Error

"detail": [

{

"loc": [

"string"

],

"msg": "string",

"type": "string"

}

500: {"description":"The server has encountered a situation it does not know how to handle."}

**POST**

**Post Request for creating a new user in database**

**URL:** <http://127.0.0.1:8000/user/signup>

**This request is used for creating a new user in database**

This endpoint lets you create a new user in database. As you provide name , email , date of birth and password fields as BODY() param. This POST request will insert a new user row in databse schema.

**Request Body ( Required ): Example**

{

"name": "Joelay field",

"dob": "2000-12-11",

"email": "joe@xyz.com",

"password": "any"

}

Response:

Code: Description

200 Successful Response

Example Schema: [

{

"name": "Joelay field",

"dob": "2000-12-11",

"email": "joe@xyz.com",

"password": "any"

}

]

422 Validation Error

"detail": [

{

"loc": [

"string"

],

"msg": "string",

"type": "string"

}

500: {"description":"The server has encountered a situation it does not know how to handle."}

### **POST**

**Post request for user login**

URL: <http://127.0.0.1:8000/user/login>

This endpoint is to get refreshed tokens for registered users.

Example Value:

{

"email": "sitaram@gmail.com",

"password": "any"

}

Responses:

Code: Description

200 Successful Response

"string"

422 Validation Error

{

"detail": [

{

"loc": [

"string"

],

"msg": "string",

"type": "string"

}

]

}

500: {"description":"The server has encountered a situation it does not know how to handle."}

**PATCH**

**Patch request for updating fields of any user in database**

URL: [http://127.0.0.1:8000/users/{user\_id}](http://127.0.0.1:8000/users/%7buser_id%7d)

**You can perform update actions for users.**

This endpoint will let you update data of users available in db. Note that email field can't be updated yet you have to provide it's value to authenticate yourself. and rest attributes will get update in db.

{Path Param} user\_id : Integer ( ID of user)

Request Body:

{

"name": "Joelay field",

"dob": "2000-12-11",

"email": "joe@xyz.com",

"password": "any"

}

Responses:

Code: Description

200 Successful Response

"string"

422 Validation Error

{

"detail": [

{

"loc": [

"string"

],

"msg": "string",

"type": "string"

}

]

}

500: {"description":"The server has encountered a situation it does not know how to handle."}

**DELETE**

**delete request for deleting any user in database**

URL: [http://127.0.0.1:8000/users/{user\_id}/del](http://127.0.0.1:8000/users/%7buser_id%7d/del)

**Delete any user by using his ID**

delete any user by passing his ID number as integer

{Path Param} user\_id : Integer ( ID of user)

Responses:

Code: Description

200 Successful Response

"string"

422 Validation Error

{

"detail": [

{

"loc": [

"string"

],

"msg": "string",

"type": "string"

}

]

}

500: {"description":"The server has encountered a situation it does not know how to handle."}