# Title: Documentation for the Likes Microservices API

## Introduction:

The Flask API application provided in this guide is designed to manage users, contents, and likes using MongoDB as the backend database. This guide will walk you through the available functions and endpoints of the application and demonstrate how to interact with the API.

## Requirements:

Before using the application, make sure you have Docker Installed

## Running the Application:

1. Download the source code of the application and save it in a directory.
2. Open a terminal or command prompt and navigate to the directory containing the source code.
3. Run the application by executing the following command:

```
docker-compose up --build
```

## API Endpoints and Functions:

### 1. Create a New User:

- Endpoint: `/user`
- Method: POST
- Request Body: JSON object with the following fields:
  - `name` (required): Name of the user.
  - `email` (required): Email address of the user.
  - `password` (required): Password of the user.

**Example:**

```
{
    "name": "John Doe",
    "email": "john@example.com",
    "password": "secretpassword"
}
```

- **Response:** If the user is successfully created, the API will respond with a JSON object containing the message and the information of the newly created user, including the assigned `_id`.

**2. Fetch All Users:**

- *Endpoint:`/user`*
- *Method: GET*
- *Response: The API will respond with a JSON array containing information about all the users stored in the database.*

**3. Create New Content:**

- *Endpoint: `/content`*
- *Method: POST*
- *Request Body: JSON object with the following fields:*
  - *`user_id` (required): The `_id` of the user who created the content.*
  - *`name` (required): Name of the content.*
  - *`desc` (required): Description of the content.*

**Example:**

```
{
    "user_id": "615c012e02a31d001fd429d2",
    "name": "New Blog Post",
    "desc": "Check out my latest blog post!"
}
```

**Response:** If the content is successfully created, the API will respond with a JSON object containing the message and information of the newly created content, including the assigned `_id`.

**4. Fetch All Contents:**

*- Endpoint: `/content`*

*- Method: GET*

*- Response:The API will respond with a JSON array containing information about all the contents stored in the database.*

**5. Like a Content:**

*- Endpoint: `/like`*

*- Method:POST*

*- Request Body: JSON object with the following fields:*

*- `user_id` (required): The `_id` of the user who liked the content.*

*- `content_id` (required): The `_id` of the content being liked.*

**Example:**

```
{
   "user_id": "615c012e02a31d001fd429d2",
   "content_id": "615c01c402a31d001fd429d4"
}
```

**- Response:** The API will respond with a JSON object containing the message indicating that the like event was stored successfully.

**6.Check if a User has Liked a Content:**

*- Endpoint: `/check_like`*

*- Method: GET*

*- Query Parameters:*

*- `user_id` (required): The `_id` of the user.*

*- `content_id` (required): The `_id` of the content.*

- **Response:** The API will respond with a JSON object containing the key `"liked"` with a boolean value (`true` if the user has liked the content, `false` otherwise).

**7. Get the Total Likes for a Content:**

- *Endpoint: `/total_likes`*
- *Method: GET*
- *Query Parameter:*
  - *`content_id` (required): The `_id` of the content.*
- **Response**: The API will respond with a JSON object containing the key `"total_likes"` with the total number of likes for the specified content.

## Usage Example:

1. Create a new user:

```
  POST /user

  Request Body:

  {
     "name": "John Doe",
     "email": "john@example.com",
     "password": "secretpassword"
  }

  Response:

  {
     "message": "user inserted successfully",
     "new user": {
        "_id": "615c012e02a31d001fd429d2",
        "name": "John Doe",
        "email": "john@example.com",
        "password": "secretpassword"
     }
  }
```

2. Fetch all users:

```
GET /user
Response:
[
    {
        "_id": "615c012e02a31d001fd429d2",
        "name": "John Doe",
        "email": "john@example.com"
    },
]
```

3. Create new content:

```
POST /content
Request Body:
{
    "user_id": "615c012e02a31d001fd429d2",
    "name": "New Blog Post",
    "desc": "Check out my latest blog post!"
}
Response:
{
    "message": "new content created successfully",
    "content_info": {
        "_id": "615c01c402a31d001fd429d4",
        "user_id": "615c012e02a31d001fd429d2",
        "name": "New Blog Post",
        "desc": "Check out my latest blog post!"
    }
}
```

4. Fetch all contents:

```
GET /content

Response:

[

    {

        "_id": "615c01c402a31d001fd429d4",

        "user_id": "615c012e02a31d001fd429d2",

        "name": "New Blog Post",

        "desc": "Check out my latest blog post!"

    },

]
```

5. Like a content:

```
POST /like

Request Body:

{

    "user_id": "615c012e02a31d001fd429d2",

    "content_id": "615c01c402a31d001fd429d4"

}

Response:

{

    "message": "Like event stored successfully"

}
```

6. Check if a user has liked a content:

```
GET /check_like?user_id=615c012e02a31d001fd429d2&content_id=615c01c402a31d001fd429d4


  Response:

  {

      "liked": true

  }
```

7. Get the total likes for a content:

```
GET /total_likes?content_id=615c01c402a31d001fd429d4
```

Response:

```
{
    "total_likes": 1
}
```