

Finstreet Customer Details APIs

Using Node.js, MySQL and JWT

HTML form:



A screenshot of a web browser showing a "User Registration" form. The browser's address bar displays "localhost:8080". The form includes five input fields: "Username", "Email address", "Password", "total orders", and "Select image to upload:". Below the "Select image to upload:" label, there is a "Choose File" button, the text "No file chosen", and a "Submit" button.



A second screenshot of the "User Registration" form, now with data entered. The browser's address bar still shows "localhost:8080". The "Username" field contains "Bhargav", "Email address" contains "bhargavtest@gmail.com", "Password" contains "*****", and "total orders" contains "2". The "Select image to upload:" section now shows a "Choose File" button, the text "form.JPG", and a "Submit" button.

On Submit -> Redirected to /users/all

```
localhost:8080/users/all

1 // 20211024015436
2 // http://localhost:8080/users/all
3
4 [
5   {
6     "user_name": "Bhargav",
7     "user_id": "14147ced-4ea4-4de6-978f-0f3697cb941e",
8     "user_email": "bhargavtest@gmail.com",
9     "user_password": "$2a$08$FWv4dCVGDD1YgFjcAIzOru2KUqOV9gye43j0mH7jncfLDmM.ZcDKm",
10    "user_image": {
11      "type": "Buffer",
12      "data": [
13        102,
14        111,
15        114,
16        109,
17        46,
18        74,
19        80,
20        71
21      ]
22    },
23    "total_orders": 2,
24    "last_logged_in": "2021-10-23T19:36:00.000Z",
25    "createdAt": "2021-10-23T19:36:00.000Z",
26    "updatedAt": "2021-10-23T19:36:00.000Z"
27  },
28  {
29    "user_name": "test",
30    "user_id": "fdc48f13-5804-42a4-8946-96a2a43ae17c",
31    "user_email": "test@gmail.com",
32    "user_password": "test",
33    "user_image": {
34      "type": "Buffer",
35      "data": [
36        104,
37        116,
38        109,
39        108,
40        46,
41        74,
42        80,
43        71
44      ]
45    },
46    "total_orders": 3,
47    "last_logged_in": "2021-10-23T19:37:02.000Z",
48    "createdAt": "2021-10-23T19:37:02.000Z",
49    "updatedAt": "2021-10-23T20:08:05.000Z"
50  }
51 ]
```

- ***/details***

Structure: BASE_URL/details/\${user_id}

Returns: object:{...user_details}

The screenshot displays a REST client interface with a dark theme. At the top, there's a header bar with 'Builder' and 'Team Library' tabs, a user profile 'Bhargav Jit...', and various utility icons. Below the header, a URL bar shows 'http://localhost:8080/' with a dropdown menu. The main area is divided into tabs: 'GET', 'Params', 'Send', and 'Save'. The 'GET' tab is active, showing the URL 'http://localhost:8080/users/details/14147ced-4ea4-4de6-978f-0f3697cb941e'. Below the URL bar, there are tabs for 'Authorization', 'Headers (1)', 'Body', 'Pre-request Script', and 'Tests'. The 'Authorization' tab is selected, showing a 'Type' dropdown set to 'No Auth'. Below the authorization section, there are tabs for 'Body', 'Cookies', 'Headers (10)', and 'Test Results'. The 'Body' tab is active, showing a JSON response in 'Pretty' format. The response is a JSON object containing user details. The status bar at the bottom right indicates 'Status: 200 OK' and 'Time: 112 ms'.

```
{
  "user_name": "Bhargav",
  "user_id": "14147ced-4ea4-4de6-978f-0f3697cb941e",
  "user_email": "bhargavtest@gmail.com",
  "user_password": "$2a$08$Fwv4dCvcGD1YgFjcAIz0ru2KUqOV9gye43j0mH7jncfLdmM.ZcDKm",
  "user_image": {
    "type": "Buffer",
    "data": [102, 111, 114, 109, 46, 74, 80, 71],
    "total_orders": 2,
    "last_logged_in": "2021-10-23T19:36:00.000Z",
    "createdAt": "2021-10-23T19:36:00.000Z",
    "updatedAt": "2021-10-23T19:36:00.000Z"
  }
}
```

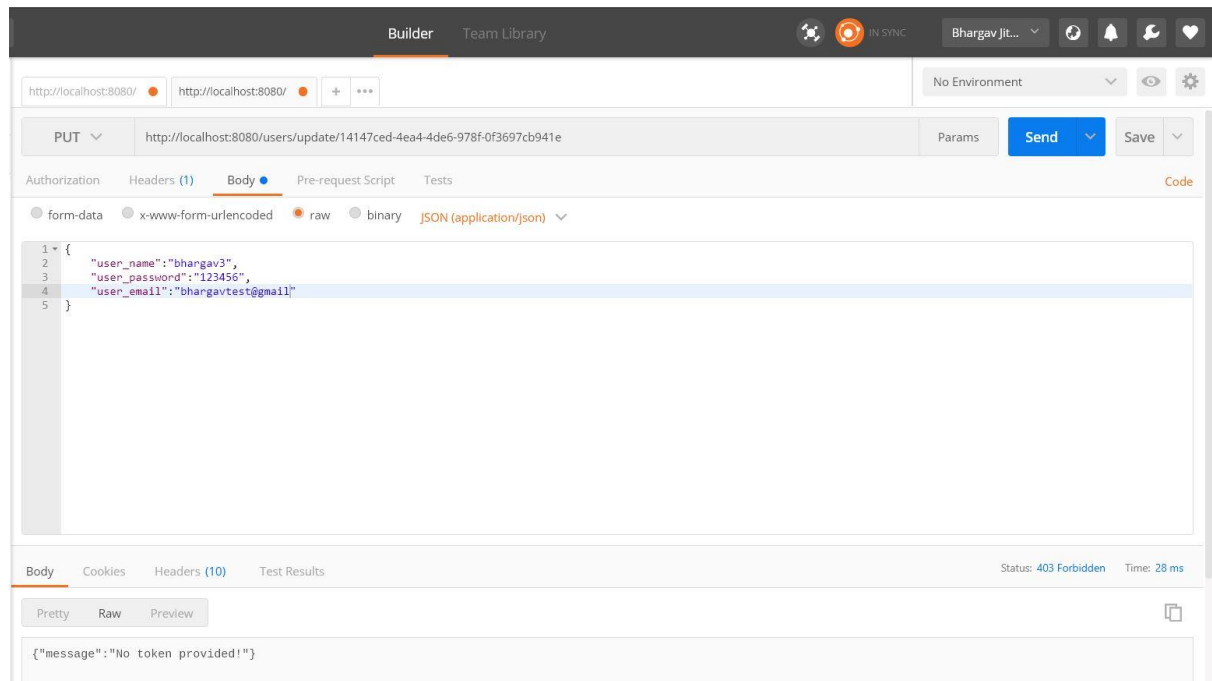
- ***/update***

Structure: BASE_URL/update

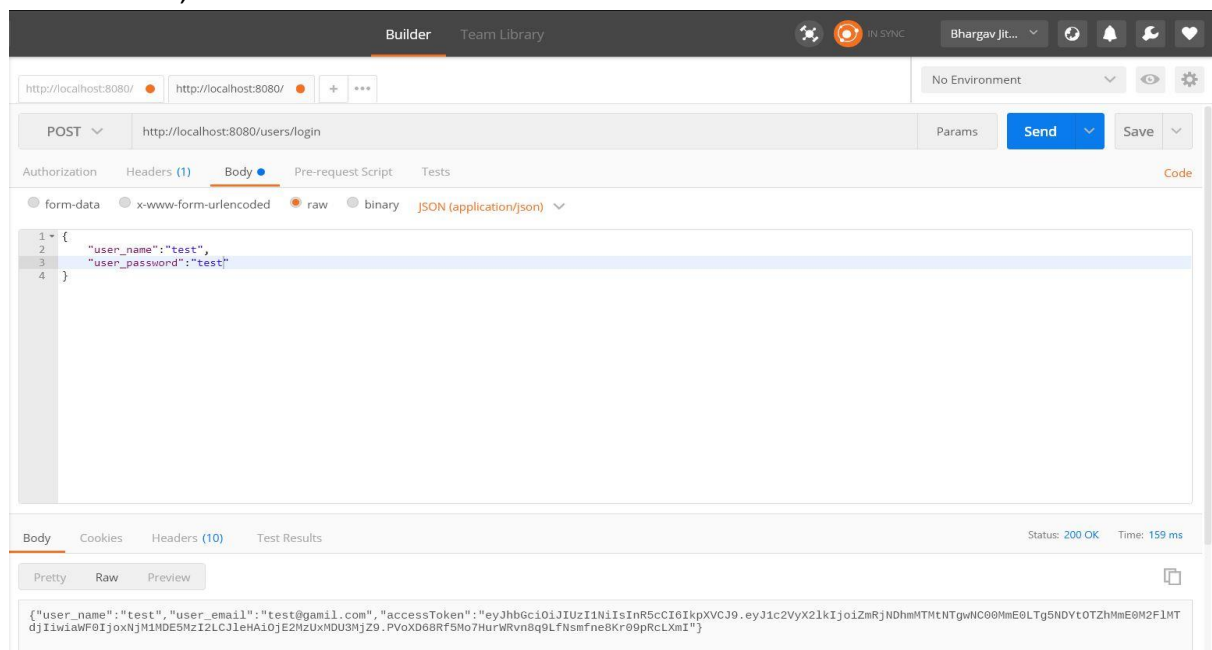
Request Body: object:{...new_details_of_user}

Returns: Some kind of a success or failure message for acknowledgement

1. Update without login: Fails with error message “No token provided!”



2. Update with Login: Using username and password generates unique access token, valid for 24hrs.



3. Adding access token to header

The screenshot shows the Postman interface for a POST request. The URL is `http://localhost:8080/users/update/fdc48f13-5804-42a4-8946-96a2a43ae17c`. The request is configured with the following headers:

Key	Value	Description
<input checked="" type="checkbox"/> Content-Type	application/json	
<input checked="" type="checkbox"/> x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkjoiZmRjNDh...	

Buttons for 'Send', 'Save', and 'Params' are visible. The 'Send' button is highlighted in blue.

4. Updating total orders

The screenshot shows the Postman interface for a PUT request. The URL is `http://localhost:8080/users/update/fdc48f13-5804-42a4-8946-96a2a43ae17c`. The request is configured with the following body (JSON):

```
1 {  
2   "user_name": "test",  
3   "user_password": "test",  
4   "user_email": "test@gmail.com",  
5   "total_orders": "3"  
6 }
```

The response is shown in the 'Body' tab, indicating a successful update:

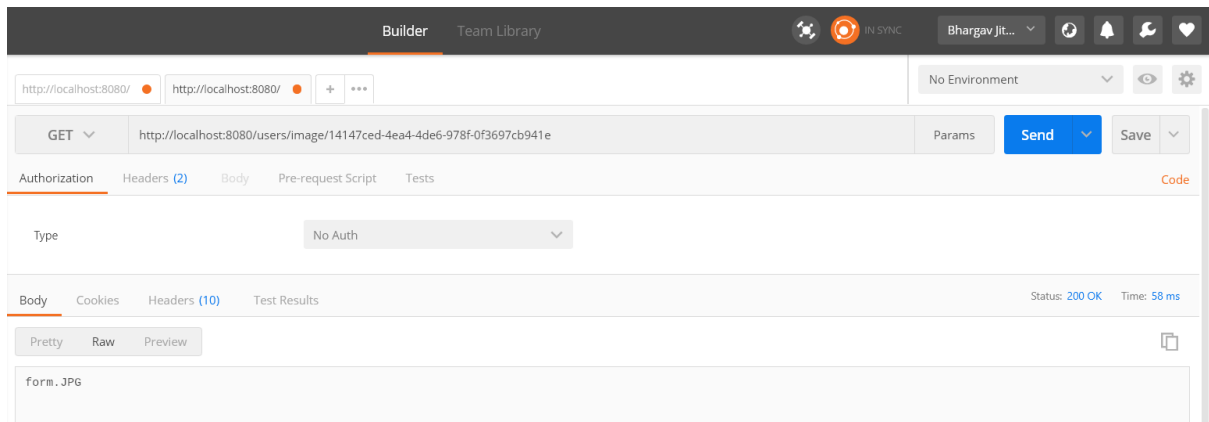
```
{ "message": "User updated successfully." }
```

The status is 200 OK and the time taken is 538 ms.

- ***/image***

Structure: BASE_URL/image/\${user_id}

Returns: object:{...user_image}



- ***/insert***

Structure: *BASE_URL/insert*

Request Body: *object:{user_details}*

Returns: *Some kind of success or failure message*

1. **Creating a new user: Get Success message**

The screenshot shows the Postman interface for a POST request to `http://localhost:8080/users/insert`. The request body is a JSON object: `{ "user_name": "test2", "user_password": "test", "user_email": "test2@gmail.com", "total_orders": "3" }`. The response status is `200 OK` with a time of `290 ms`. The response body is `successfully inserted new User`.

2. **Adding user using duplicate user_name: Error: "Username already in use"**

The screenshot shows the Postman interface for a POST request to `http://localhost:8080/users/insert` with the same JSON body as in the first screenshot. The response status is `400 Bad Request` with a time of `47 ms`. The response body is `{"message": "Failed! Username is already in use!"}`.

- ***/delete***

Structure: BASE_URL/delete/\${user_id}

Returns: A success or failure message

