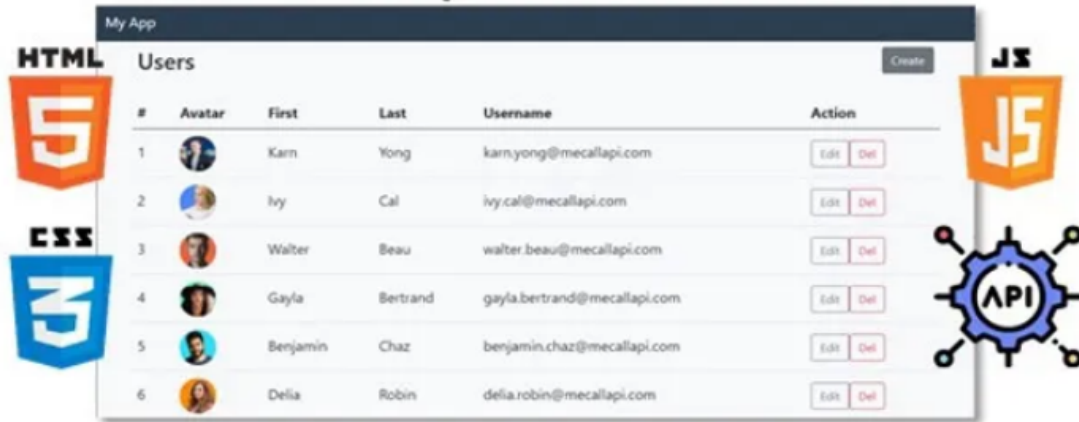


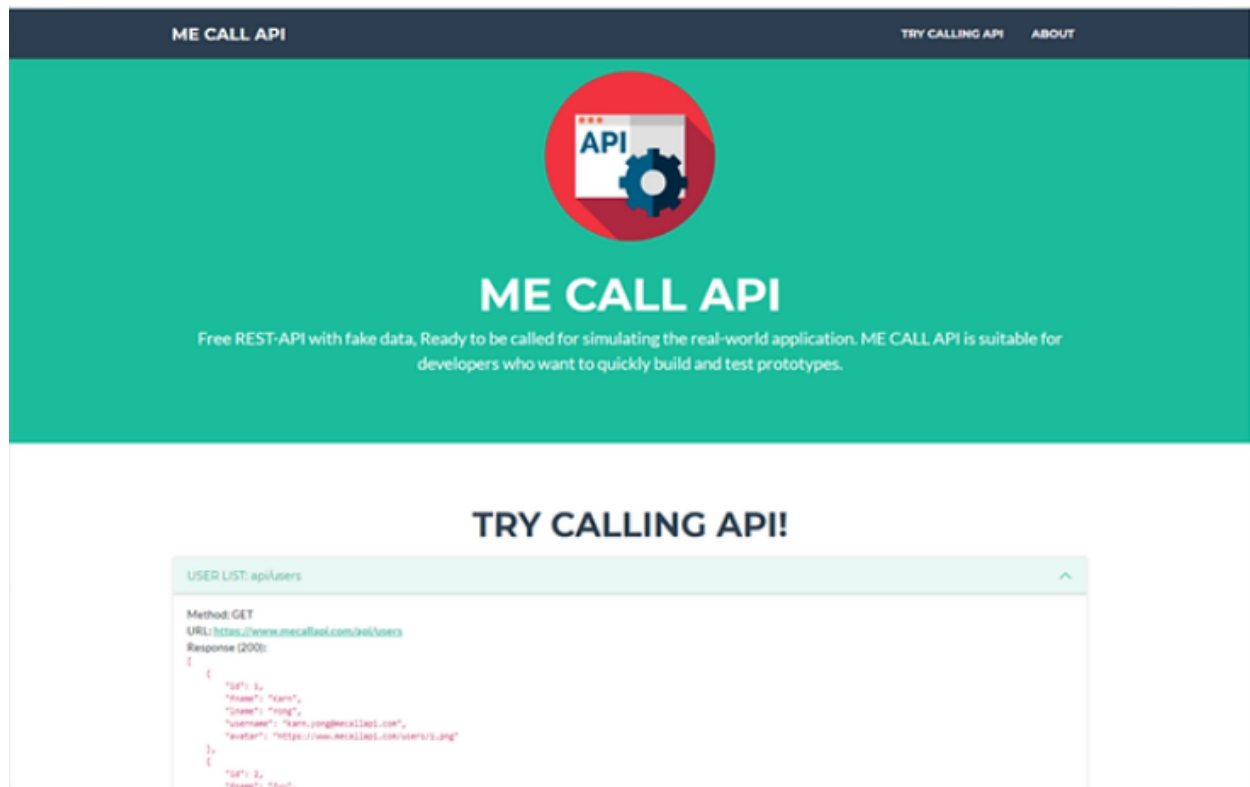
Basic HTML, CSS, JS (Bootstrap 5) for CRUD operations with API



In this article, we will give you a tutorial for creating a web application with just only basic HTML, CSS and JavaScript (based on Bootstrap 5) to perform CRUD operations. Well, CRUD operations are the four basic operations of manipulating data including Create/Construct, Read, Update and Delete.

Furthermore, our CRUD operations will perform by the use of an external API from MeCallAPI.com. If you want to try a mockup API for CRUD and authentication operations, feel free to check on the website.

Users						Create
#	Avatar	First	Last	Username	Action	
1		Karn	Yong	karn.yong@mecallapi.com	Edit	Del
2		Ivy	Cal	ivy.cal@mecallapi.com	Edit	Del
3		Walter	Beau	walter.beau@mecallapi.com	Edit	Del
4		Gayla	Bertrand	gayla.bertrand@mecallapi.com	Edit	Del
5		Benjamin	Chaz	benjamin.chaz@mecallapi.com	Edit	Del
6		Delia	Robin	delia.robin@mecallapi.com	Edit	Del
7		Hector	Graves	hector.graves@mecallapi.com	Edit	Del
8		Diego	Greene	diego.greene@mecallapi.com	Edit	Del
9		Izzie	Mooney	izzie.mooney@mecallapi.com	Edit	Del
10		Romeo	Garner	romeo.garner@mecallapi.com	Edit	Del



Software Installation

You only need a Text Editor/IDE (VS Code, Notepad, etc.) and a web browser (Chrome, Firefox, Edge, etc.) to do this tutorial!

Let's Code! (HTML and CSS)

Starting for the HTML which are documents designed to be displayed in a web browser. There are more than a hundred of HTML elements you can choose to put on your HTML file and make the magic happen.

Create **index.html** with the following links:

- [Bootstrap 5](#) framework to make your life easier to create a responsive web (line 9 and 54)
- [Sweetalert](#) for easily creating nice popups using JavaScript (line 53)
- **index.css** for defining extra CSS (Cascading Style Sheets) to style your **index.html** in addition from the Bootstrap (line 13)
- **index.js** for JavaScript using in **index.html** to call API for CRUD operations (line 52)

```

<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
      scale=1">

    <!-- Bootstrap CSS -->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.m
in.css" rel="stylesheet" integrity="sha384-
+0n0xVW2eSR50omGNYDnhzAbDsOXxcvSN1TPprVMTNDbiYZCxYbOO17+AMvyTG2x"
crossorigin="anonymous">

    <title>CRUD FROM API</title>

    <link href="index.css" rel="stylesheet">
  </head>

```

```

<body>

  <nav class="navbar navbar-dark bg-mynav">
    <div class="container-fluid">
      <a class="navbar-brand" href="#">My App</a>
    </div>
  </nav>

  <div class="container">
    <div class="d-flex bd-highlight mb-3">
      <div class="me-auto p-2 bd-highlight"><h2>Users</h2>
      <div class="p-2 bd-highlight">
        <button type="button" class="btn btn-secondary"
onclick="showUserCreateBox()">Create</button>
      </div>
    </div>
  </div>

```

```

<div class="table-responsive">
  <table class="table">
    <thead>
      <tr>
        <th scope="col">#</th>
        <th scope="col">Avatar</th>
        <th scope="col">First</th>
        <th scope="col">Last</th>
        <th scope="col">Username</th>
        <th scope="col">Action</th>
      </tr>
    </thead>
    <tbody id="mytable">
      <tr>
        <th scope="row" colspan="5">Loading...</th>
      </tr>
    </tbody>
  </table>
</div>
</div>

```

```

<script src="index.js"></script>
<script src="https://cdn.jsdelivr.net/npm/sweetalert2
  @11.0.16/dist/sweetalert2.all.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap
  @5.0.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-
  gtEjrD/SeCtmISkJKNUaaKMoLD0//ElJ19smozuHV6z3Iehds+3U1b9Bn9P1x0x4"
  crossorigin="anonymous"><
  /script>
</body>
</html>

```

Create **index.css** to define extra CSS (Cascading Style Sheets) for your **index.html** in addition from the CSS provided by Bootstrap.

```
.bg-mynav {  
    background-color:#2c3e50;  
}  
  
body {  
    font-size:1.25rem;  
    background-color:#f6f8fa;  
}  
  
td {  
    line-height:3rem;  
}
```

Read operation (JavaScript)

Create file **index.js** to call an **API** for **CRUD** operations starting from **Read**.

API URL: <https://www.mecallapi.com/api/users>

Method: GET

To read data with JavaScript, we will use **AJAX (Asynchronous JavaScript And XML)** technique to have our web site update asynchronously. Concretely, we will have data exchange using **API** in the background. Thus, the data is retrieved, we will have our web page updated without doing a refresh.

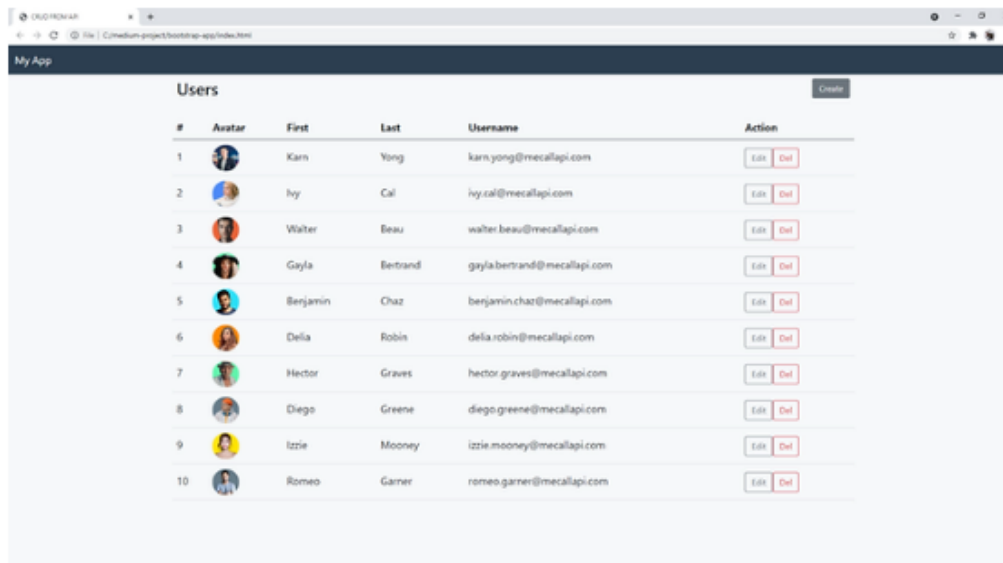
We will use **XMLHttpRequest** to call an **API** for retrieving data in **JSON** and display/update such data on the web page. To update the data on the web page, we will manipulate **HTML DOM (Document Object Model)**. In our code, we will update the data in table by referring to the **id** of the **HTML** data table element, namely (**mytable**) in line 21.











```

function loadTable(){
  const xhttp = new XMLHttpRequest();
  xhttp.open("GET","https://www.mecallapi.com/api/users");
  xhttp.send();
  xhttp.onreadystatechange=function(){
    if(this.readyState == 4&&this.status==200){
      console.log(this.responseText);
      vartrHTML='';
      constobjects=JSON.parse(this.responseText);
      for(letobjectofobjects){
        trHTML+<tr>;
        trHTML+<td>'+object['id']+'</td>';
        trHTML+<td></td>';
        trHTML+<td>'+object['fname']+'</td>';
        trHTML+<td>'+object['lname']+'</td>';
        trHTML+<td>'+object['username']+'</td>';
        trHTML+<td><button type="button" class="btn btn-
outline-secondary"
onclick="showUserEditBox('+object['id']+')">
Edit</button>';
        trHTML+<button type="button" class="btn btn-outline-
danger" onclick="userDelete('+object['id']+')">
Del</button></td>';
        trHTML+<td>';
      }
      document.getElementById("mytable").innerHTML=trHTML;
    }
  };
}
loadTable();

```

Open index.html on a web Browser, you will see the result:



#	Avatar	First	Last	Username	Action
1		Kam	Yong	kam.yong@mecallapi.com	<button>Edit</button> <button>Del</button>
2		Ivy	Cal	ivy.cal@mecallapi.com	<button>Edit</button> <button>Del</button>
3		Walter	Beau	walter.beau@mecallapi.com	<button>Edit</button> <button>Del</button>
4		Gayla	Bertrand	gayla.bertrand@mecallapi.com	<button>Edit</button> <button>Del</button>
5		Benjamin	Chaz	benjamin.chaz@mecallapi.com	<button>Edit</button> <button>Del</button>
6		Della	Robin	della.robin@mecallapi.com	<button>Edit</button> <button>Del</button>
7		Hector	Graves	hector.graves@mecallapi.com	<button>Edit</button> <button>Del</button>
8		Diego	Greene	diego.greene@mecallapi.com	<button>Edit</button> <button>Del</button>
9		Izzie	Mooney	izzie.mooney@mecallapi.com	<button>Edit</button> <button>Del</button>
10		Romeo	Garner	romeo.garner@mecallapi.com	<button>Edit</button> <button>Del</button>

Data from the API showing

Create operation (JavaScript)

API URL: <https://www.mecallapi.com/api/users/create>

Method: POST

Sample body (JSON):

```
{
  "fname": "Cat",
  "lname": "Chat",
  "username": "cat.chat@mecallapi.com",
  "email": "cat.chat@mecallapi.com",
  "avatar": "https://www.mecallapi.com/users/cat.png"
}
```

Add the following code in index.js

```
function showUserCreateBox(){
  Swal.fire({
    title: 'Create user',
    html: '
      <input id="id" type="hidden">'+
      <input id="fname" class="swal2-input"
        placeholder="First">'+
      <input id="lname" class="swal2-input"
        placeholder="Last">'+
      <input id="username" class="swal2-input"
        placeholder="Username">'+
      <input id="email" class="swal2-input"
        placeholder="Email">',
    focusConfirm: false,
    preConfirm: ()=>{
      userCreate();
    }
  })
}

function userCreate(){
  const fname=document.getElementById("fname").value;
  const lname=document.getElementById("lname").value;
  const username=document.getElementById("username").value;
  const email=document.getElementById("email").value;
```

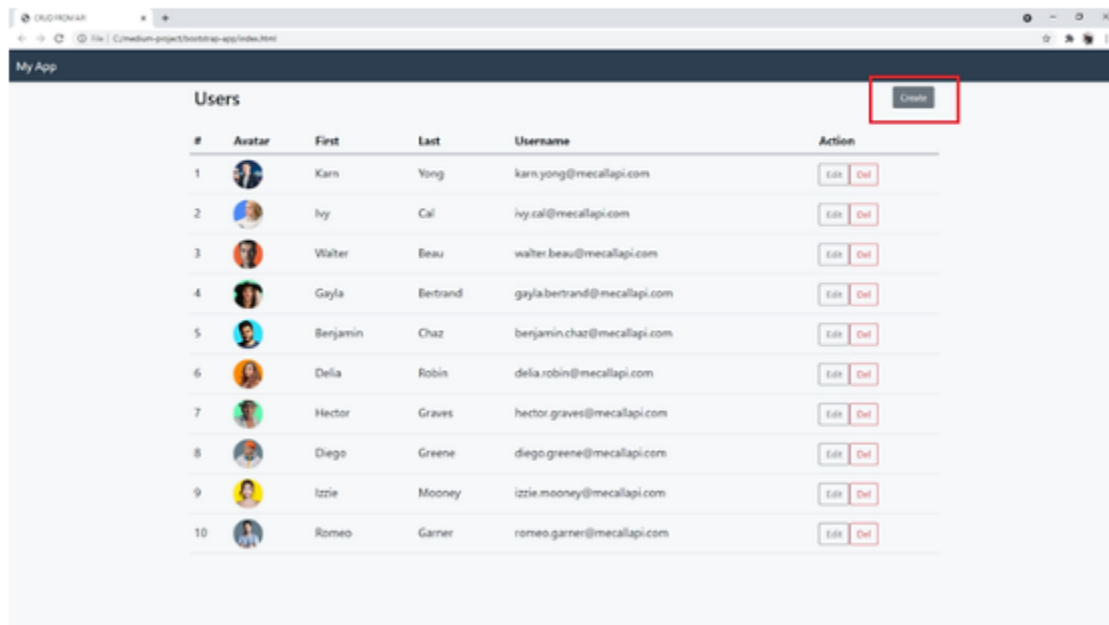
```

const xhttp = new XMLHttpRequest();
xhttp.open("POST", "https://www.mecallapi.com/api/users/create");
xhttp.setRequestHeader("Content-Type", "application/json;
    charset=UTF-8");
xhttp.send(JSON.stringify({
    "fname": fname, "lname": lname, "username": username, "email": email,
    "avatar": "https://www.mecallapi.com/users/cat.png"
}));

xhttp.onreadystatechange = function(){
    if(this.readyState == 4&&this.status==200){
        const objects = JSON.parse(this.responseText);
        Swal .fire(objects['message']);
        loadTable();
    }
};
}

```

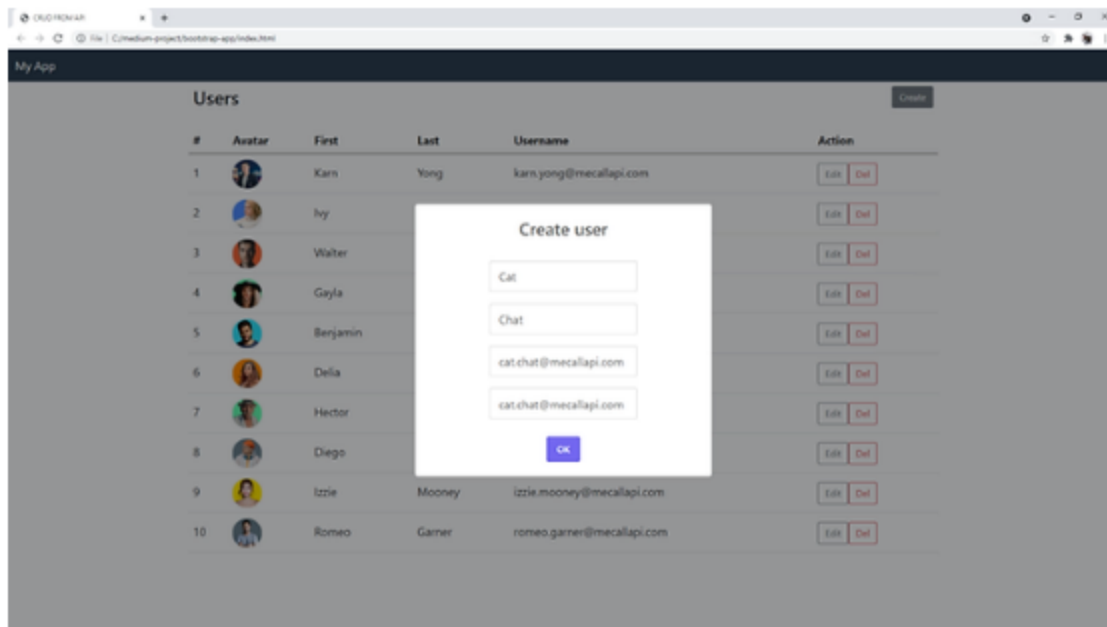
Refresh index.html and try to perform the create operation:



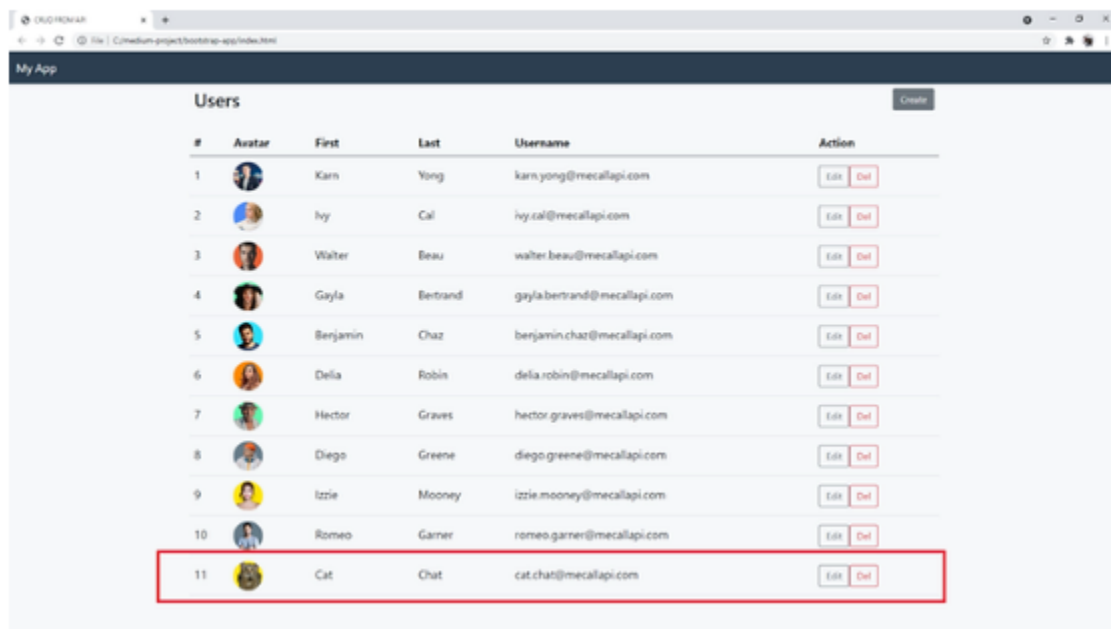
The screenshot shows a web browser window with the address bar displaying 'C:\medium-project\bootstrap-app\index.html'. The page title is 'My App'. Below the title bar, there is a 'Users' section. In the top right corner of this section, there is a 'Create' button highlighted with a red rectangle. Below the button is a table with 10 rows of user data. Each row has columns for '#', 'Avatar', 'First', 'Last', 'Username', and 'Action'. The 'Action' column contains 'Edit' and 'Del' buttons for each user.

#	Avatar	First	Last	Username	Action
1		Kam	Yong	kam.yong@mecallapi.com	Edit Del
2		Ivy	Cal	ivy.cal@mecallapi.com	Edit Del
3		Walter	Beau	walter.beau@mecallapi.com	Edit Del
4		Gayla	Bertrand	gayla.bertrand@mecallapi.com	Edit Del
5		Benjamin	Chaz	benjamin.chaz@mecallapi.com	Edit Del
6		Delia	Robin	delia.robin@mecallapi.com	Edit Del
7		Hector	Graves	hector.graves@mecallapi.com	Edit Del
8		Diego	Greene	diego.greene@mecallapi.com	Edit Del
9		Izzie	Mooney	izzie.mooney@mecallapi.com	Edit Del
10		Romeo	Garner	romeo.garner@mecallapi.com	Edit Del

Click Create



Input data



New data is added

UPDATE operation (JavaScript)

API URL: <https://www.mecallapi.com/api/users/update>

Method: PUT

Sample body (JSON):

```
{
  "id": 11,
  "fname": "Cat",
  "lname": "Gato",
  "username": "cat.gato@mecallapi.com",
  "email": "cat.gato@mecallapi.com",
  "avatar": "https://www.mecallapi.com/users/cat.png"
}
```

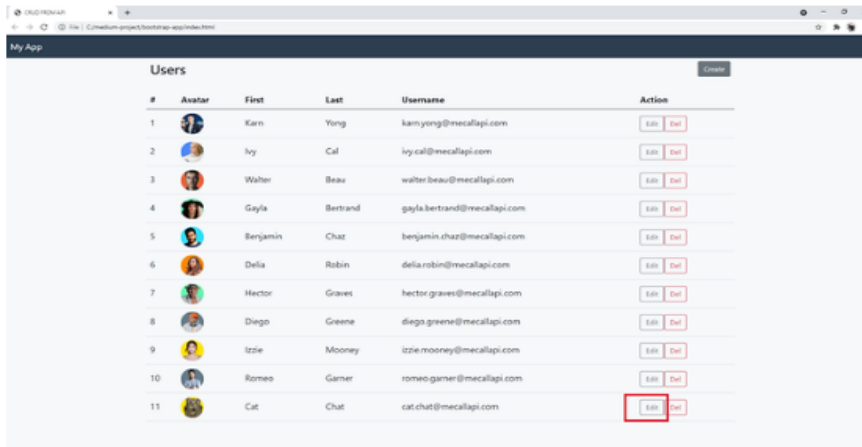
Add the following code in index.js

```
function showUserEditBox(id){
  console.log(id);
  const xhttp = new XMLHttpRequest();
  xhttp.open("GET","https://www.mecallapi.com/api/users/"+id);
  xhttp.send();
  xhttp.onreadystatechange = function(){
    if(this.readyState==4&&this.status==200){
      const objects=JSON.parse(this.responseText);
      const user=objects['user'];
      console.log(user);
      Swal.fire({
        title: 'Edit User',
        html: `
          <input id="id" type="hidden" value='${user['id']}'>'+
          <input id="fname" class="swal2-input"
            placeholder="First" value='${user['fname']}'>'+
          <input id="lname" class="swal2-input"
            placeholder="Last" value='${user['lname']}'>'+
          <input id="username" class="swal2-input"
            placeholder="Username" value='${user['username']}'>'+
          <input id="email" class="swal2-input"
            placeholder="Email" value='${user['email']}'>'+
          focusConfirm: false,
          preConfirm: ()=>{
            userEdit();
          }
        }
      });
    }
  };
}

function userEdit(){
  const id=document.getElementById("id").value;
  const fname=document.getElementById("fname").value;
  const lname=document.getElementById("lname").value;
  const username=document.getElementById("username").value;
  const email=document.getElementById("email").value;

  const xhttp = new XMLHttpRequest();
  xhttp.open("PUT","https://www.mecallapi.com/api/users/update");
  xhttp.setRequestHeader("Content-Type","application/json; charset=UTF-8");
  xhttp.send(JSON.stringify({
    "id": id,"fname": fname,"lname": lname,"username":
      username,"email": email,
    "avatar": "https://www.mecallapi.com/users/cat.png"
  }));
  xhttp.onreadystatechange = function(){
    if(this.readyState == 4&&this.status==200){
      const tobjects=JSON.parse(this.responseText);
      Swal.fire(tobject['message']);
      loadTable();
    }
  };
}
```

Refresh `index.html` and try to perform the update operation:

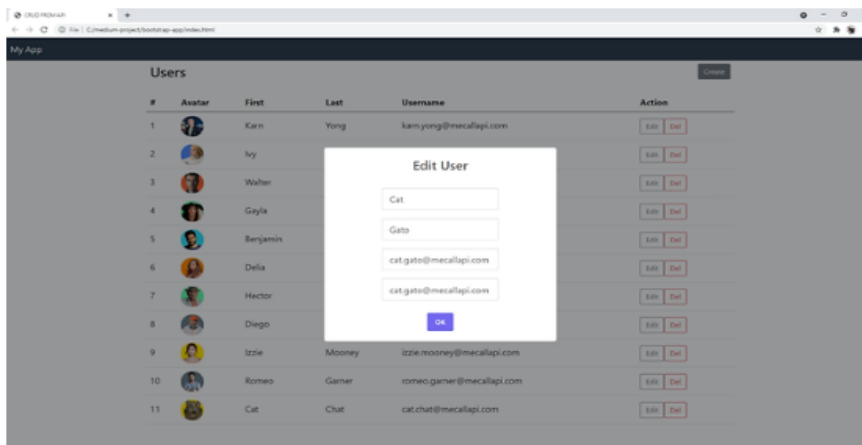


My App

Users Create

#	Avatar	First	Last	Username	Action
1		Kam	Yong	kam.yong@mecallapi.com	Edit Del
2		Ivy	Cal	ivy.cal@mecallapi.com	Edit Del
3		Walter	Beau	walter.beau@mecallapi.com	Edit Del
4		Gayla	Bertrand	gayla.bertrand@mecallapi.com	Edit Del
5		Benjamin	Chat	benjamin.chat@mecallapi.com	Edit Del
6		Della	Robin	della.robin@mecallapi.com	Edit Del
7		Hector	Graves	hector.graves@mecallapi.com	Edit Del
8		Diego	Greene	diego.greene@mecallapi.com	Edit Del
9		Izzie	Mooney	izzie.mooney@mecallapi.com	Edit Del
10		Romeo	Garner	romeo.garner@mecallapi.com	Edit Del
11		Cat	Chat	cat.chat@mecallapi.com	Edit Del

Edit the newly added user



My App

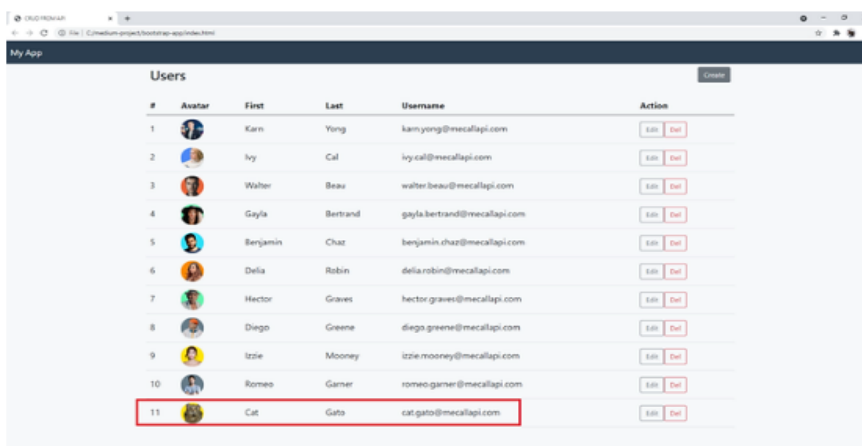
Users Create

Edit User

OK

#	Avatar	First	Last	Username	Action
1		Kam	Yong	kam.yong@mecallapi.com	Edit Del
2		Ivy	Cal	ivy.cal@mecallapi.com	Edit Del
3		Walter	Beau	walter.beau@mecallapi.com	Edit Del
4		Gayla	Bertrand	gayla.bertrand@mecallapi.com	Edit Del
5		Benjamin	Chat	benjamin.chat@mecallapi.com	Edit Del
6		Della	Robin	della.robin@mecallapi.com	Edit Del
7		Hector	Graves	hector.graves@mecallapi.com	Edit Del
8		Diego	Greene	diego.greene@mecallapi.com	Edit Del
9		Izzie	Mooney	izzie.mooney@mecallapi.com	Edit Del
10		Romeo	Garner	romeo.garner@mecallapi.com	Edit Del
11		Cat	Chat	cat.chat@mecallapi.com	Edit Del

Input data



My App

Users Create

#	Avatar	First	Last	Username	Action
1		Kam	Yong	kam.yong@mecallapi.com	Edit Del
2		Ivy	Cal	ivy.cal@mecallapi.com	Edit Del
3		Walter	Beau	walter.beau@mecallapi.com	Edit Del
4		Gayla	Bertrand	gayla.bertrand@mecallapi.com	Edit Del
5		Benjamin	Chat	benjamin.chat@mecallapi.com	Edit Del
6		Della	Robin	della.robin@mecallapi.com	Edit Del
7		Hector	Graves	hector.graves@mecallapi.com	Edit Del
8		Diego	Greene	diego.greene@mecallapi.com	Edit Del
9		Izzie	Mooney	izzie.mooney@mecallapi.com	Edit Del
10		Romeo	Garner	romeo.garner@mecallapi.com	Edit Del
11		Cat	Gato	cat.gato@mecallapi.com	Edit Del

Data is updated

DELETE operation (JavaScript)

API URL: <https://www.mecallapi.com/api/users/delete>

Method: DELETE

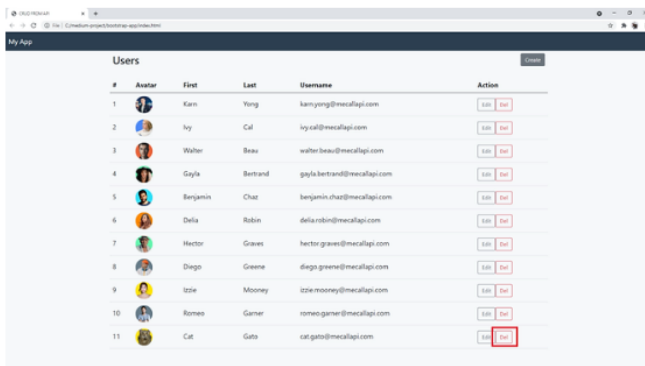
Sample Body (JSON):

```
{
  "id": 11
}
```

Add the following code in index.js

```
function userDelete(id){
  const xhttp = new XMLHttpRequest();
  xhttp.open("DELETE", "https://www.mecallapi.com/api/users/delete");
  xhttp.setRequestHeader("Content-Type", "application/json");
  xhttp.send(JSON.stringify({
    "id": id
  }));
  xhttp.onreadystatechange = function(){
    if(this.readyState==4){
      const objects = JSON.parse(this.responseText);
      Swal.fire(objects['message']);
      loadTable();
    }
  };
}
```

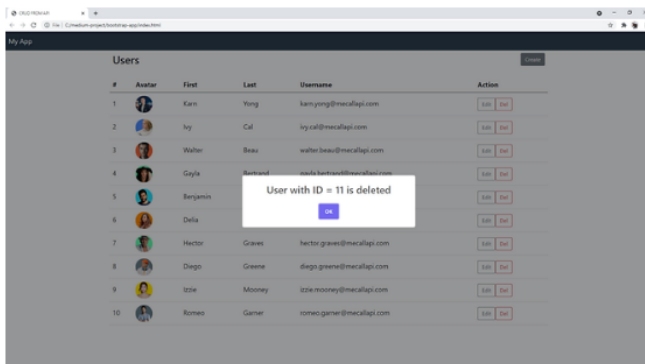
Refresh index.html and try to perform the delete operation:



The screenshot shows a web application titled "My App" with a "Users" section. It contains a table with 11 users. The user with ID 11, named "Cat" with the email "cat.gato@mecallapi.com", is highlighted with a red box. The "Action" column for each user has "Edit" and "Delete" buttons.

#	Avatar	First	Last	Username	Action
1		Kari	Yong	kari.yong@mecallapi.com	<button>Edit</button> <button>Delete</button>
2		Ivy	Cal	ivy.cal@mecallapi.com	<button>Edit</button> <button>Delete</button>
3		Walter	Beau	walter.beau@mecallapi.com	<button>Edit</button> <button>Delete</button>
4		Gayle	Bertrand	gayle.bertrand@mecallapi.com	<button>Edit</button> <button>Delete</button>
5		Benjamin	Chaz	benjamin.chaz@mecallapi.com	<button>Edit</button> <button>Delete</button>
6		Della	Rubin	della.rubin@mecallapi.com	<button>Edit</button> <button>Delete</button>
7		Hector	Graves	hector.graves@mecallapi.com	<button>Edit</button> <button>Delete</button>
8		Diego	Greene	diego.greene@mecallapi.com	<button>Edit</button> <button>Delete</button>
9		Izzy	Mooney	izzy.mooney@mecallapi.com	<button>Edit</button> <button>Delete</button>
10		Romeo	Garner	romeo.garner@mecallapi.com	<button>Edit</button> <button>Delete</button>
11		Cat	Gato	cat.gato@mecallapi.com	<button>Edit</button> <button>Delete</button>

Delete the newly added user



The screenshot shows the same "Users" table as before, but with a confirmation popup displayed over it. The popup says "User with ID = 11 is deleted" and has an "OK" button. The user with ID 11 is still highlighted with a red box.

#	Avatar	First	Last	Username	Action
1		Kari	Yong	kari.yong@mecallapi.com	<button>Edit</button> <button>Delete</button>
2		Ivy	Cal	ivy.cal@mecallapi.com	<button>Edit</button> <button>Delete</button>
3		Walter	Beau	walter.beau@mecallapi.com	<button>Edit</button> <button>Delete</button>
4		Gayle	Bertrand	gayle.bertrand@mecallapi.com	<button>Edit</button> <button>Delete</button>
5		Benjamin	Chaz	benjamin.chaz@mecallapi.com	<button>Edit</button> <button>Delete</button>
6		Della	Rubin	della.rubin@mecallapi.com	<button>Edit</button> <button>Delete</button>
7		Hector	Graves	hector.graves@mecallapi.com	<button>Edit</button> <button>Delete</button>
8		Diego	Greene	diego.greene@mecallapi.com	<button>Edit</button> <button>Delete</button>
9		Izzy	Mooney	izzy.mooney@mecallapi.com	<button>Edit</button> <button>Delete</button>
10		Romeo	Garner	romeo.garner@mecallapi.com	<button>Edit</button> <button>Delete</button>
11		Cat	Gato	cat.gato@mecallapi.com	<button>Edit</button> <button>Delete</button>

Popup is showing

My App

Users Create

#	Avatar	First	Last	Username	Action
1		Kam	Yong	kam.yong@mecallapi.com	Edit Delete
2		Ivy	Cal	ivy.cal@mecallapi.com	Edit Delete
3		Walter	Beau	walter.beau@mecallapi.com	Edit Delete
4		Gayla	Bertrand	gayla.bertrand@mecallapi.com	Edit Delete
5		Benjamin	Chaz	benjamin.chaz@mecallapi.com	Edit Delete
6		Delia	Robin	delia.robin@mecallapi.com	Edit Delete
7		Hector	Graves	hector.graves@mecallapi.com	Edit Delete
8		Diego	Greene	diego.greene@mecallapi.com	Edit Delete
9		Izzie	Mooney	izzie.mooney@mecallapi.com	Edit Delete
10		Romeo	Garner	romeo.garner@mecallapi.com	Edit Delete

Data is deleted