

This Flask app is a simple user management system that allows you to:

View all users

View a single user by ID

Add a new user

Update a user's details

Delete a user

It uses an in-memory list to store users (no database), and handles all the logic through different URL routes (/users, /users/<id>). The app responds with JSON data and supports frontend interaction using CORS. The main page is rendered using an index.html file. It's great for learning how basic web APIs and CRUD operations work using Python and Flask.

The screenshot shows a code editor interface with the following details:

- File Explorer:** On the left, it lists files and folders:
 - OPEN EDITORS:** index.html, script.js, styles.css, app.py
 - CRUD_APPLICATION:** .vscode, static, templates, index.html
- Code Editor:** The main area displays the `app.py` file content, which defines a Flask application for a CRUD API. It includes routes for listing users, getting a user by ID, and creating a new user.
- Bottom Bar:** Includes icons for search, file operations, and system status (language, battery, time).

```
1 from flask import Flask, render_template, jsonify, request
2 from flask_cors import CORS
3
4 app = Flask(__name__)
5 CORS(app)
6
7 # In-memory data store
8 users = [
9     {"id": 1, "name": "Gift", "age": 25},
10    {"id": 2, "name": "Lewis", "age": 27},
11    {"id": 3, "name": "John", "age": 28}
12 ]
13
14 @app.route("/")
15 def index():
16     return render_template("index.html")
17
18 @app.route("/users", methods=["GET"])
19 def get_users():
20     return jsonify(users), 200
21
22 @app.route("/users/<int:user_id>", methods=["GET"])
23 def get_user(user_id):
24     user = next((u for u in users if u["id"] == user_id), None)
25     if user:
26         return jsonify(user), 200
27     return jsonify({"error": "User not found"}), 404
28
29 @app.route("/users", methods=["POST"])
30 def create_user():
31     data = request.get_json()
32     if not data or "name" not in data or "age" not in data:
33         return jsonify({"error": "Name and age are required"}), 400
34
35     new_id = max([u["id"] for u in users], default=0) + 1
36     new_user = {"id": new_id, "name": data["name"], "age": data["age"]}
37     users.append(new_user)
```

The screenshot shows a Windows desktop environment with a Visual Studio Code (VS Code) window open. The title bar of the VS Code window reads "crud_application". The left sidebar shows a file tree with "OPEN EDITORS" containing "index.html", "script.js", "# styles.css", and "app.py". The "CRUD_APPLICATION" folder contains "vscode", "static", "templates", and "index.html". The "app.py" file is the active editor, displaying Python code for a Flask-based CRUD application. The code defines routes for creating, updating, and deleting users, and handles the main application logic. The status bar at the bottom of the VS Code window shows "Ln 1, Col 1" and "Python 3.12.2 64-bit". The taskbar at the bottom of the screen displays various pinned icons, including Microsoft Edge, File Explorer, Task View, Task Manager, and several other application icons.

```
30 def create_user():
31     new_user = request.get_json()
32     users.append(new_user)
33     return jsonify(new_user), 201
34
35 @app.route("/users/<int:user_id>", methods=["PUT"])
36 def update_user(user_id):
37     data = request.get_json()
38     user = next((u for u in users if u["id"] == user_id), None)
39     if not user:
40         return jsonify({"error": "User not found"}), 404
41
42     user["name"] = data.get("name", user["name"])
43     user["age"] = data.get("age", user["age"])
44     return jsonify(user), 200
45
46 @app.route("/users/<int:user_id>", methods=["DELETE"])
47 def delete_user(user_id):
48     global users
49     user = next((u for u in users if u["id"] == user_id), None)
50     if not user:
51         return jsonify({"error": "User not found"}), 404
52
53     users = [u for u in users if u["id"] != user_id]
54     return jsonify({"message": "User deleted"}), 200
55
56 if __name__ == "__main__":
57     app.run(debug=True)
58
59
```

Summary of What the HTML Code Does:

This HTML file is the user interface for a Flask-based CRUD application. It allows users to:

Add a new user with a name and age

Edit existing user information

Delete users from the list

Search for users by name

Display all users in a table format

The frontend connects to the backend (app.py) using JavaScript (from script.js) and dynamically updates the content without refreshing the page.

✨ Few Important Code Sections:

Search Field and Button

Allows the user to search for names:

```
<input type="text" id="searchInput" placeholder="Search by name...">
```

```
<button id="searchBtn">Search</button>
```

User Form (Add/Edit)

A form that captures name and age:

```
<form id="userForm">
  <input type="hidden" id="userId" value="">
  <input type="text" id="name" required>
  <input type="number" id="age" min="1" required>
</form>
```

Users Table

Displays all users and their actions:

```
<table>
  <thead>
    <tr>
      <th>ID</th>
      <th>Name</th>
      <th>Age</th>
      <th>Actions</th>
    </tr>
  </thead>
  <tbody id="userTable"></tbody>
</table>
```

Alert Message Box

Used to show success or error messages:

```
<div id="alertBox" class="alert"></div>
```

CSS & JS Linking

Links to external styles and scripts:

```
<link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
<script src="{{ url_for('static', filename='script.js') }}></script>
```

The screenshot shows the VS Code interface with the file 'index.html' open in the editor. The code defines a container div containing a search form and a user form. It includes input fields for name and age, and buttons for submit and cancel. Below the forms is a heading 'Users' followed by a table definition.

```
1 python <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>CRUD APP</title>
6     <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}>
7 </head>
8 <body>
9     <div class="container">
10         <h1>CRUD App</h1>
11
12         <div id="alertBox" class="alert"></div>
13
14         <div class="search-container">
15             <input type="text" id="searchInput" placeholder="Search by name...">
16             <button id="searchBtn">Search</button>
17         </div>
18
19         <div class="user-form">
20             <form id="userForm">
21                 <input type="hidden" id="userId" value="">
22                 <div class="form-group">
23                     <label for="name">Name</label>
24                     <input type="text" id="name" required>
25                 </div>
26                 <div class="form-group">
27                     <label for="age">Age</label>
28                     <input type="number" id="age" min="1" required>
29                 </div>
30                 <button type="submit" id="submitBtn">Add User</button>
31                 <button type="button" id="cancelBtn" class="cancel" style="display: none;">Cancel</button>
32             </form>
33         </div>
34
35         <h2>Users</h2>
36         <table>
37             <thead>
38                 <tr>
39                     <th>ID</th>
40                     <th>Name</th>
41                     <th>Age</th>
42                     <th>Actions</th>
43                 </tr>
44             </thead>
45             <tbody id="userTable"></tbody>
46         </table>
47     </div>
48
49     <script src="{{ url_for('static', filename='script.js') }}></script>
50 </body>
51 </html>
```

The screenshot shows the VS Code interface with the file 'index.html' open. The code now includes a CSS link and a script tag to load 'script.js'. The table definition has been updated to include a tbody element.

```
1 python <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>CRUD APP</title>
6     <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}>
7 </head>
8 <body>
9     <div class="container">
10         <h1>CRUD App</h1>
11
12         <div id="alertBox" class="alert"></div>
13
14         <div class="search-container">
15             <input type="text" id="searchInput" placeholder="Search by name...">
16             <button id="searchBtn">Search</button>
17         </div>
18
19         <div class="user-form">
20             <form id="userForm">
21                 <input type="hidden" id="userId" value="">
22                 <div class="form-group">
23                     <label for="name">Name</label>
24                     <input type="text" id="name" required>
25                 </div>
26                 <div class="form-group">
27                     <label for="age">Age</label>
28                     <input type="number" id="age" min="1" required>
29                 </div>
30                 <button type="submit" id="submitBtn">Add User</button>
31                 <button type="button" id="cancelBtn" class="cancel" style="display: none;">Cancel</button>
32             </form>
33         </div>
34
35         <h2>Users</h2>
36         <table>
37             <thead>
38                 <tr>
39                     <th>ID</th>
40                     <th>Name</th>
41                     <th>Age</th>
42                     <th>Actions</th>
43                 </tr>
44             </thead>
45             <tbody id="userTable"></tbody>
46         </table>
47     </div>
48
49     <script src="{{ url_for('static', filename='script.js') }}></script>
50 </body>
51 </html>
```

This CSS file styles the CRUD App interface to make it clean, readable, and visually appealing. It:

Sets a light background and modern font across the app.

Centers the content using a styled .container.

Styles inputs and buttons to be user-friendly.

Adds hover effects for buttons to improve interactivity.

Formats the table for displaying user data neatly.

Defines alert message styles for success and error feedback.

Few Important Code Sections (Core Styling):

Centering and Styling the App Container:

```
.container {  
    max-width: 800px;  
    margin: auto;  
    background-color: #fff;  
    padding: 20px;  
    border-radius: 8px;  
}
```

Input and Button Styling:

```
input, button {  
    padding: 10px;  
    margin: 5px;  
    font-size: 14px;  
    border-radius: 4px;  
    border: 1px solid #ccc;  
}
```

Hover Effect for Buttons:

```
button:hover {  
    transform: scale(1.05);  
    background-color: #45a049;  
}
```

Error and Success Alert Boxes:

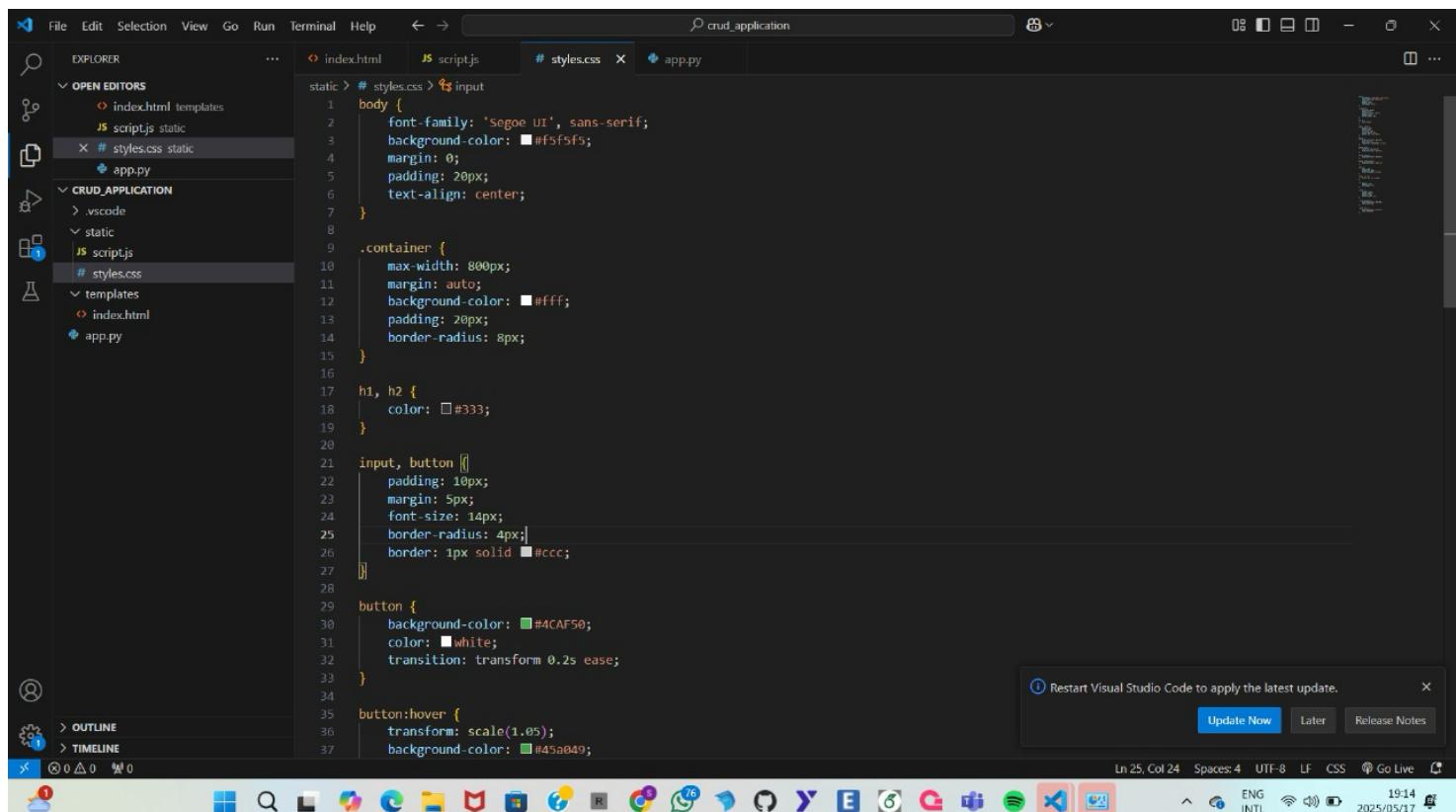
```
.alert-success {  
    background-color: #dff0d8;  
    color: #3c763d;  
}
```

```
.alert-error {  
    background-color: #f2dede;  
    color: #a94442;  
}  
}
```

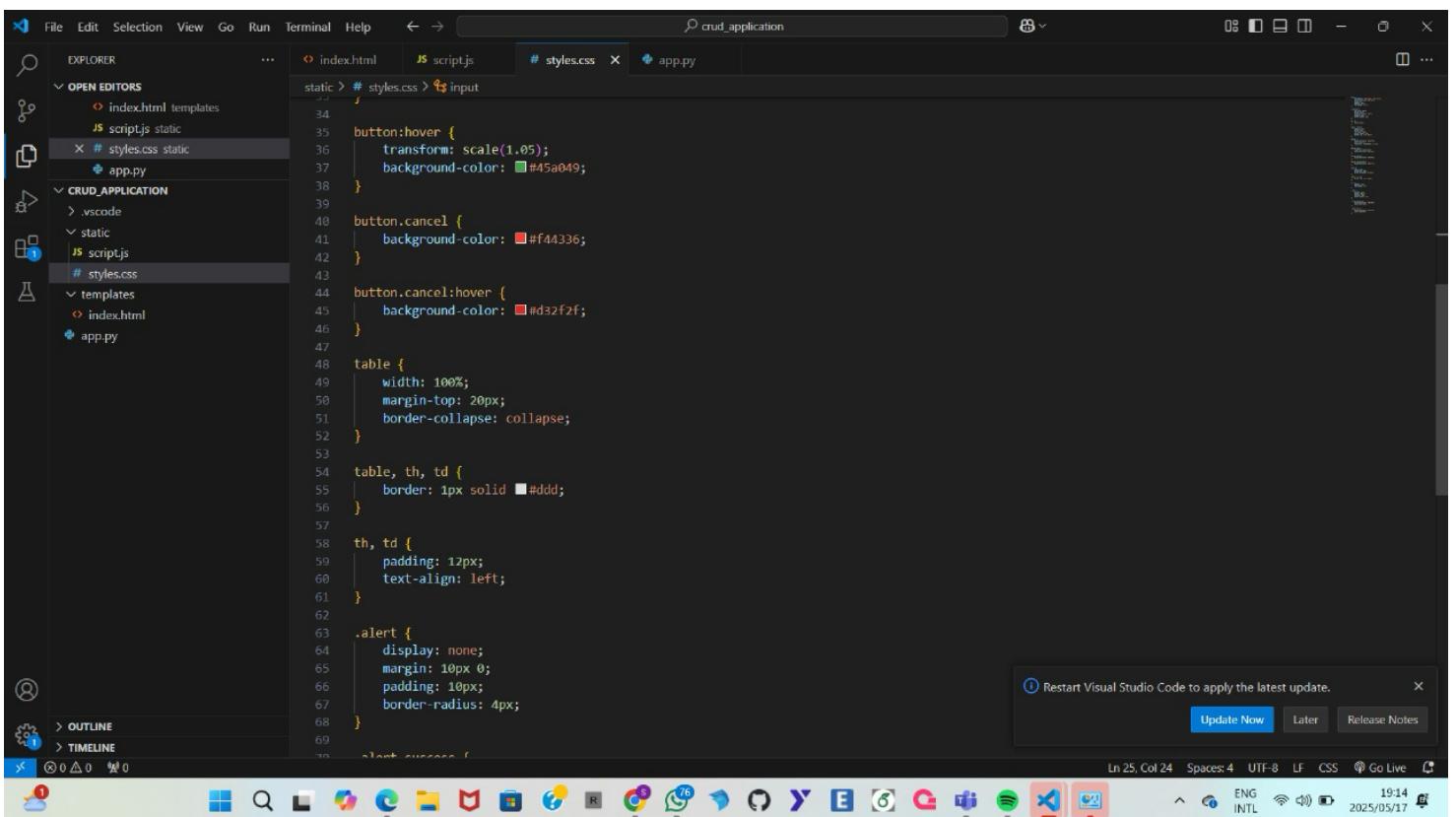
Table Styling for User List:

```
table {  
    width: 100%;  
    margin-top: 20px;  
    border-collapse: collapse;  
}  
}
```

```
table, th, td {  
    border: 1px solid #ddd;  
}  
}
```



```
static > # styles.css > input  
1  body {  
2      font-family: 'segoe ui', sans-serif;  
3      background-color: #f5f5f5;  
4      margin: 0;  
5      padding: 20px;  
6      text-align: center;  
7  }  
8  
9  .container {  
10     max-width: 800px;  
11     margin: auto;  
12     background-color: #fff;  
13     padding: 20px;  
14     border-radius: 8px;  
15  }  
16  
17  h1, h2 {  
18      color: #333;  
19  }  
20  
21  input, button {  
22      padding: 10px;  
23      margin: 5px;  
24      font-size: 14px;  
25      border-radius: 4px;  
26      border: 1px solid #ccc;  
27  }  
28  
29  button {  
30      background-color: #4CAF50;  
31      color: white;  
32      transition: transform 0.2s ease;  
33  }  
34  
35  button:hover {  
36      transform: scale(1.05);  
37      background-color: #45a049;
```



```
static > # styles.css > input
34 button:hover {
35     transform: scale(1.05);
36     background-color: #45a049;
37 }
38 }

39 button.cancel {
40     background-color: #f44336;
41 }
42 }

43 button.cancel:hover {
44     background-color: #d32f2f;
45 }
46 }

47 table {
48     width: 100%;
49     margin-top: 20px;
50     border-collapse: collapse;
51 }
52 }

53 th, td {
54     border: 1px solid #ddd;
55 }
56 }

57 th, td {
58     padding: 12px;
59     text-align: left;
60 }
61 }

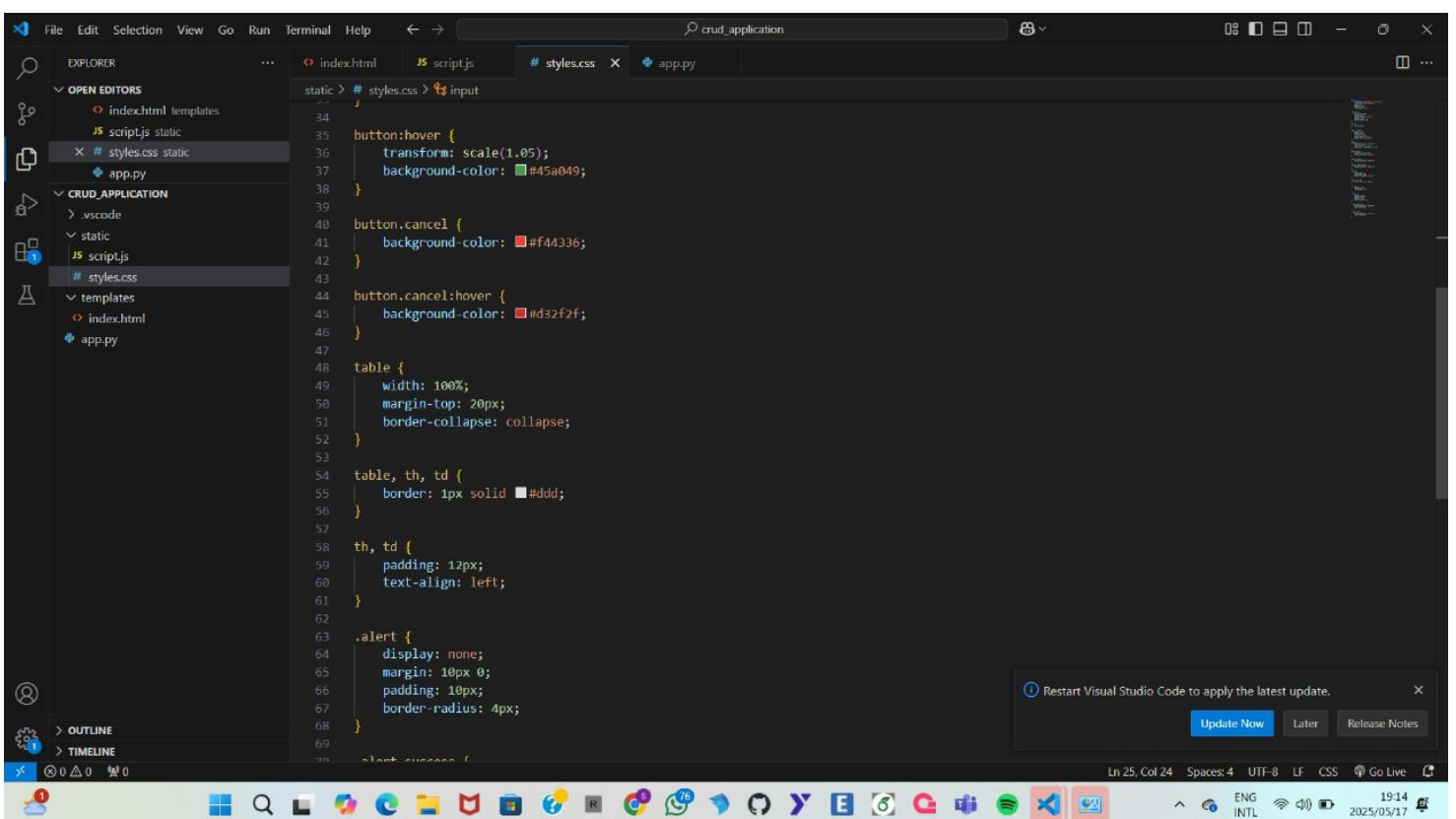
62 .alert {
63     display: none;
64     margin: 10px 0;
65     padding: 10px;
66     border-radius: 4px;
67 }
68 }

69 alert-success {
70 }
```

Restart Visual Studio Code to apply the latest update.

Update Now Later Release Notes

Ln 25, Col 24 Spaces: 4 UTF-8 LF CSS Go Live



```
static > # styles.css > input
34 button:hover {
35     transform: scale(1.05);
36     background-color: #45a049;
37 }
38 }

39 button.cancel {
40     background-color: #f44336;
41 }
42 }

43 button.cancel:hover {
44     background-color: #d32f2f;
45 }
46 }

47 table {
48     width: 100%;
49     margin-top: 20px;
50     border-collapse: collapse;
51 }
52 }

53 th, td {
54     border: 1px solid #ddd;
55 }
56 }

57 th, td {
58     padding: 12px;
59     text-align: left;
60 }
61 }

62 .alert {
63     display: none;
64     margin: 10px 0;
65     padding: 10px;
66     border-radius: 4px;
67 }
68 }

69 alert-success {
70 }
```

Restart Visual Studio Code to apply the latest update.

Update Now Later Release Notes

Ln 25, Col 24 Spaces: 4 UTF-8 LF CSS Go Live

This JavaScript file powers the interactivity of the CRUD App by:

Fetching users from a backend API and displaying them in a table.

Adding new users via a form.

Editing existing users by filling the form with their data.

Updating user information using a PUT request.

Deleting users from the list using a DELETE request.

Searching users by name and filtering the displayed list.

Showing alert messages for actions like add, update, or delete.

It dynamically updates the user list without reloading the page, making the app responsive and smooth.

Few Core/Important Code Sections

Initial Fetch of All Users:

```
function fetchUsers() {
  fetch(API_URL + '/users')
    .then(res => res.json())
    .then(data => {
      allUsers = data;
      renderUsers(data);
    });
}
```

Render Users into HTML Table:

```
function renderUsers(users) {
  userTable.innerHTML = '';
  users.forEach(user => {
    const row = document.createElement('tr');
    row.innerHTML = `
      <td>${user.id}</td>
      <td>${user.name}</td>
      <td>${user.age}</td>
      <td>
        <button onclick="editUser(${user.id})">Edit</button>
        <button onclick="deleteUser(${user.id})" class="cancel">Delete</button>
      </td>
    `;
    userTable.appendChild(row);
  });
}
```

```
});  
}  
}
```

Handle Add/Update User Form Submission:

```
userForm.addEventListener('submit', function (e) {  
    e.preventDefault();  
    const user = { name: nameField.value, age: parseInt(ageField.value) };  
    const id = userIdField.value;  
  
    const method = id ? 'PUT' : 'POST';  
    const url = id ? `${API_URL}/users/${id}` : `${API_URL}/users`;  
  
    fetch(url, {  
        method: method,  
        headers: { 'Content-Type': 'application/json' },  
        body: JSON.stringify(user)  
    })  
        .then(res => res.json())  
        .then(() => {  
            fetchUsers();  
            userForm.reset();  
            userIdField.value = '';  
            submitBtn.textContent = 'Add User';  
            cancelBtn.style.display = 'none';  
            showAlert(`User ${id ? 'updated' : 'added'} successfully!`, 'success');  
        });  
});
```

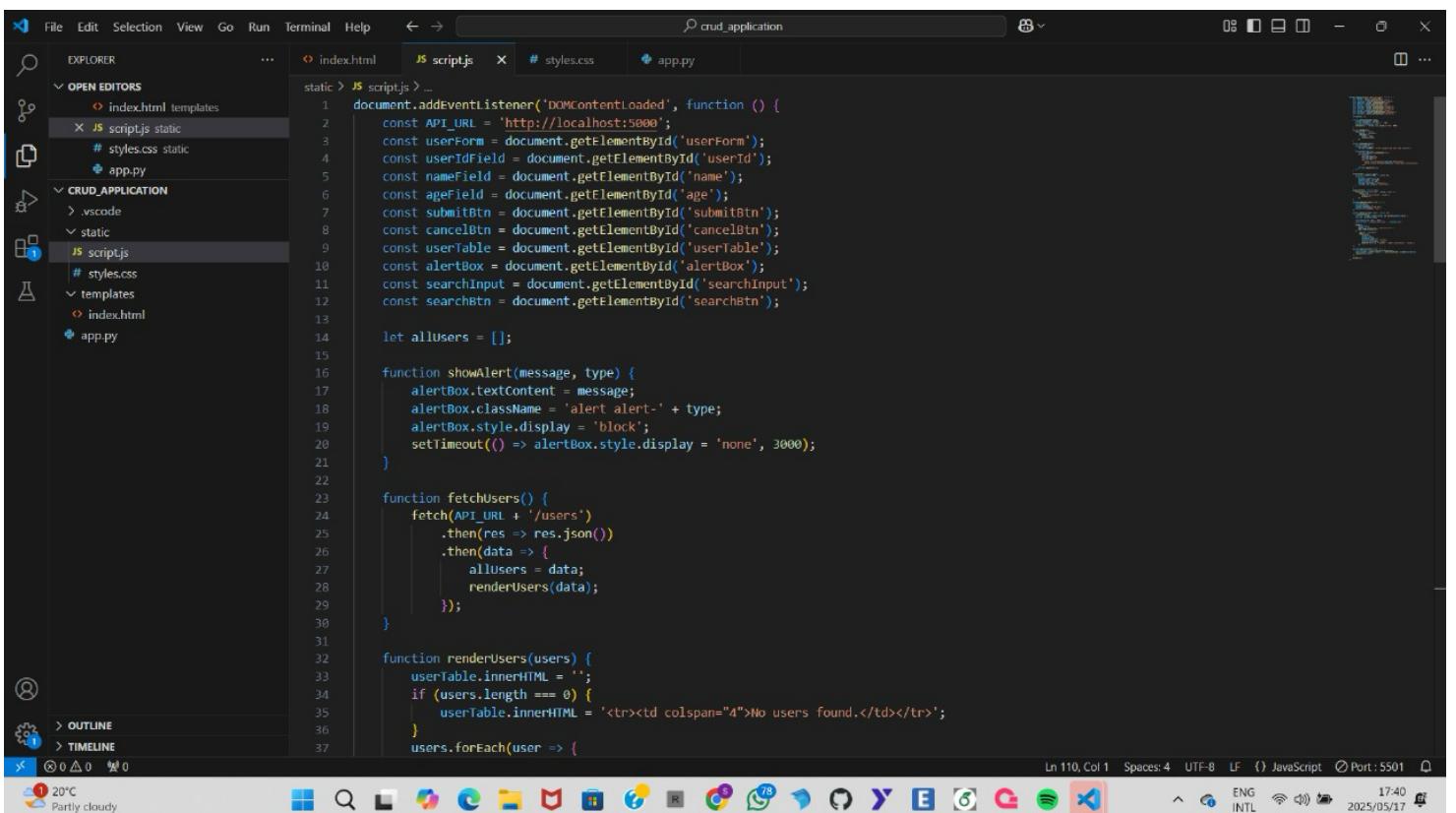
Search by Name:

```
searchBtn.addEventListener('click', () => {  
    const searchTerm = searchInput.value.toLowerCase();  
    const filtered = allUsers.filter(u => u.name.toLowerCase().includes(searchTerm));  
    renderUsers(filtered);  
});
```

Alert Notification Box:

```
function showAlert(message, type) {  
    alertBox.textContent = message;  
    alertBox.className = 'alert alert-' + type;  
    alertBox.style.display = 'block';  
    setTimeout(() => alertBox.style.display = 'none', 3000);
```

{



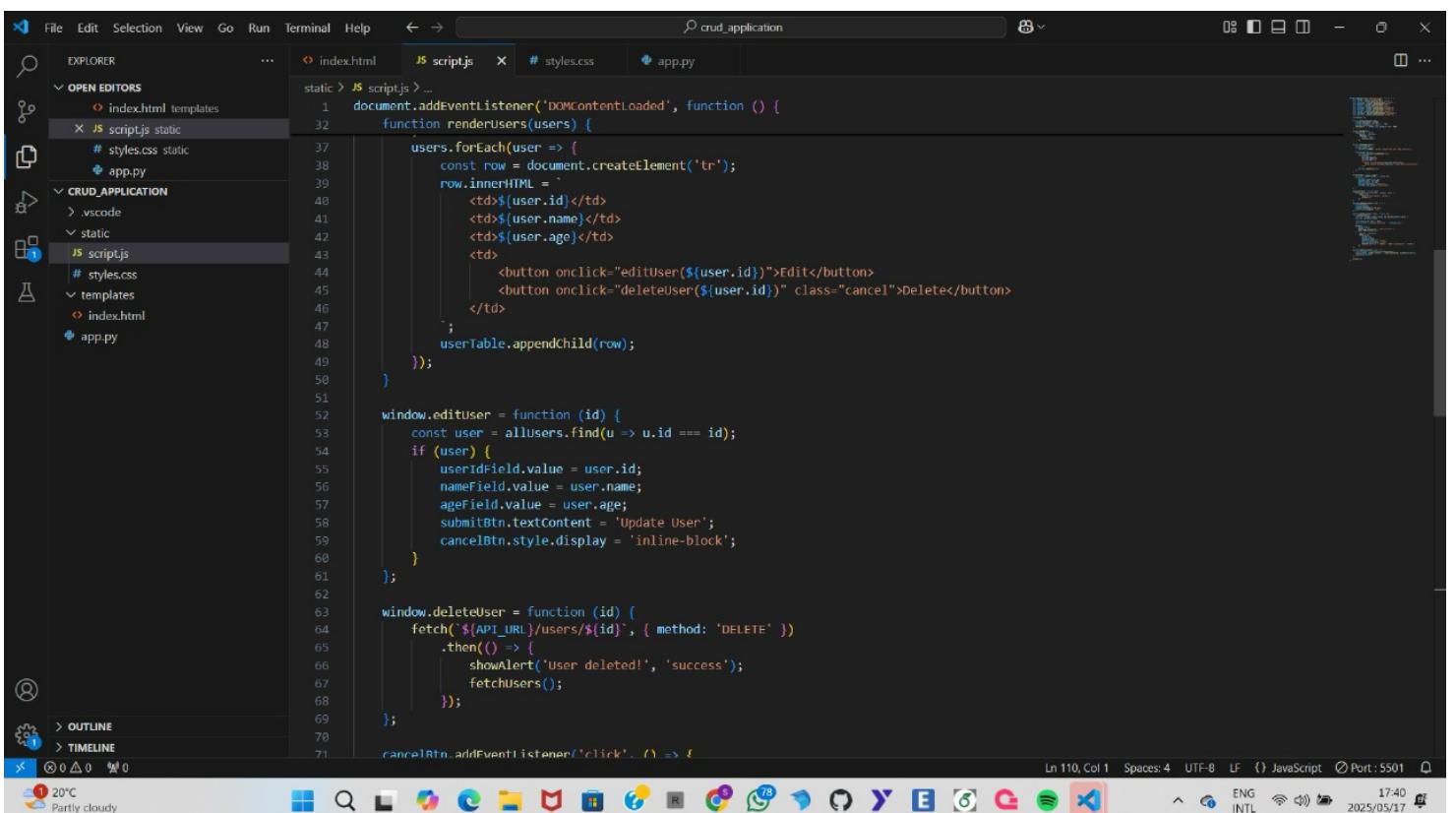
```

File Edit Selection View Go Run Terminal Help < > crud_application
OPEN EDITORS index.html JS script.js # styles.css app.py
static > JS script.js > ...
1 document.addEventListener('DOMContentLoaded', function () {
2     const API_URL = 'http://localhost:5000';
3     const userForm = document.getElementById('userForm');
4     const userIdField = document.getElementById('userId');
5     const nameField = document.getElementById('name');
6     const ageField = document.getElementById('age');
7     const submitBtn = document.getElementById('submitBtn');
8     const cancelBtn = document.getElementById('cancelBtn');
9     const userTable = document.getElementById('userTable');
10    const alertBox = document.getElementById('alertBox');
11    const searchInput = document.getElementById('searchInput');
12    const searchBtn = document.getElementById('searchBtn');
13
14    let allUsers = [];
15
16    function showAlert(message, type) {
17        alertBox.textContent = message;
18        alertBox.className = 'alert alert-' + type;
19        alertBox.style.display = 'block';
20        setTimeout(() => alertBox.style.display = 'none', 3000);
21    }
22
23    function fetchUsers() {
24        fetch(API_URL + '/users')
25            .then(res => res.json())
26            .then(data => {
27                allUsers = data;
28                renderUsers(data);
29            });
30    }
31
32    function renderUsers(users) {
33        userTable.innerHTML = '';
34        if (users.length === 0) {
35            userTable.innerHTML = '<tr><td colspan="4">No users found.</td></tr>';
36        }
37        users.forEach(user => {

```

Ln 110, Col 1 Spaces: 4 UTF-8 LF {} JavaScript Port: 5501

20°C Partly cloudy



```

File Edit Selection View Go Run Terminal Help < > crud_application
OPEN EDITORS index.html JS script.js # styles.css app.py
static > JS script.js > ...
1 document.addEventListener('DOMContentLoaded', function () {
2     function renderUsers(users) {
3         users.forEach(user => {
4             const row = document.createElement('tr');
5             row.innerHTML =
6                 <td>${user.id}</td>
7                 <td>${user.name}</td>
8                 <td>${user.age}</td>
9                 <td>
10                     <button onclick="editUser(${user.id})>Edit</button>
11                     <button onclick="deleteUser(${user.id})>Delete</button>
12                 </td>
13             ;
14             userTable.appendChild(row);
15         });
16     }
17
18     window.editUser = function (id) {
19         const user = allUsers.find(u => u.id === id);
20         if (user) {
21             userIdField.value = user.id;
22             nameField.value = user.name;
23             ageField.value = user.age;
24             submitBtn.textContent = 'Update User';
25             cancelBtn.style.display = 'inline-block';
26         }
27     };
28
29     window.deleteUser = function (id) {
30         fetch(`${API_URL}/users/${id}`, { method: 'DELETE' })
31             .then(() => {
32                 showAlert('User deleted!', 'success');
33                 fetchUsers();
34             });
35     };
36
37     cancelBtn.addEventListener('click', () => {

```

Ln 110, Col 1 Spaces: 4 UTF-8 LF {} JavaScript Port: 5501

20°C Partly cloudy

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure with files: index.html, script.js (selected), styles.css, and app.py.
- Code Editor:** Displays the script.js code. The code handles user form submission and search functionality, interacting with an API endpoint \${API_URL}/users.
- Status Bar:** Shows the current file is script.js, the line count is Ln 110, column 1, and the port is Port: 5501.
- System Tray:** Shows the weather as 20°C Partly cloudy.

```
static > JS script.js > ...
1  document.addEventListener('DOMContentLoaded', function () {
2    ...
3    cancelBtn.addEventListener('click', () => {
4      userForm.reset();
5      userIdField.value = '';
6      submitBtn.textContent = 'Add User';
7      cancelBtn.style.display = 'none';
8    });
9
10 userForm.addEventListener('submit', function (e) {
11   e.preventDefault();
12   const user = { name: nameField.value, age: parseInt(ageField.value) };
13   const id = userIdField.value;
14
15   const method = id ? 'PUT' : 'POST';
16   const url = id ? `${API_URL}/users/${id}` : `${API_URL}/users`;
17
18   fetch(url, {
19     method: method,
20     headers: { 'Content-Type': 'application/json' },
21     body: JSON.stringify(user)
22   })
23     .then(res => res.json())
24     .then(() => {
25       fetchUsers();
26       userForm.reset();
27       userIdField.value = '';
28       submitBtn.textContent = 'Add User';
29       cancelBtn.style.display = 'none';
30       showAlert(`User ${id ? 'updated' : 'added'} successfully!`, 'success');
31     });
32
33   searchBtn.addEventListener('click', () => {
34     const searchTerm = searchInput.value.toLowerCase();
35     const filtered = allUsers.filter(u => u.name.toLowerCase().includes(searchTerm));
36
37     renderUsers(filtered);
38   });
39
40   fetchUsers();
41 });
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure with files: index.html, script.js (selected), styles.css, and app.py.
- Code Editor:** Displays the script.js code. The code handles user form submission and search functionality, interacting with an API endpoint \${API_URL}/users.
- Status Bar:** Shows the current file is script.js, the line count is Ln 110, column 1, and the port is Port: 5501.
- System Tray:** Shows the weather as 20°C Partly cloudy.

```
static > JS script.js > ...
1  document.addEventListener('DOMContentLoaded', function () {
2    userForm.addEventListener('submit', function (e) {
3      ...
4      .then(() => {
5        showAlert(`User ${id ? 'updated' : 'added'} successfully!`, 'success');
6      });
7
8    searchBtn.addEventListener('click', () => {
9      const searchTerm = searchInput.value.toLowerCase();
10     const filtered = allUsers.filter(u => u.name.toLowerCase().includes(searchTerm));
11
12     renderUsers(filtered);
13   });
14
15   fetchUsers();
16 });
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
```