

JAVASCRIPT

CRUD :

```
1 function fetch(){
2   return new Promise((resolve,reject)=>{
3     setTimeout(()=>{
4       console.log("data fetching...");
5       resolve([
6         {
7           "id": 1,
8           "name": "Alice Johnson",
9           "email": "alice.johnson@example.com",
10          "age": 29,
11          "location": "New York",
12          "skills": ["JavaScript", "React", "Node.js"]
13        },
14        {
15          "id": 2,
16          "name": "Bob Smith",
17          "email": "bob.smith@example.com",
18          "age": 34,
19          "location": "San Francisco",
20          "skills": ["Python", "Django", "Machine Learning"]
21        }
22      ])
23    },1000)
24  })
25 }
26
27 let pr=fetch();
28 pr.then((val)=>{
29   console.log(val);
30 })
```

```
27 let pr=fetch();
28 pr.then((val)=>{
29   console.log(val);
30 })
```

data fetching...

```
▼ (5) [{...}, {...}, {...}, {...}, {...}] i
  ► 0: {id: 1, name: 'Alice Johnson', email: 'alice.johnson@example.com', age: 29, location: 'New York', ...}
  ► 1: {id: 2, name: 'Bob Smith', email: 'bob.smith@example.com', age: 34, location: 'San Francisco', ...}
  ► 2: {id: 3, name: 'Carol Williams', email: 'carol.williams@example.com', age: 27, location: 'Chicago', ...}
  ► 3: {id: 4, name: 'David Brown', email: 'david.brown@example.com', age: 42, location: 'Seattle', ...}
  ► 4: {id: 5, name: 'Eve Davis', email: 'eve.davis@example.com', age: 31, location: 'Austin', ...}
  length: 5
  ► [[Prototype]]: Array(0)
```

```

27   async function print(){
28       try{
29           const data=await fetch();
30           console.log(data);
31       }
32       catch(error){
33           console.log(error);
34       }
35   }
36
37   print()

```

```

data fetching...
▼ (2) [{...}, {...}] ⓘ
  ▶ 0: {id: 1, name: 'Alice Johnson', email: 'alice.johnson@example.com', age: 29, location: 'New York', ...}
  ▶ 1: {id: 2, name: 'Bob Smith', email: 'bob.smith@example.com', age: 34, location: 'San Francisco', ...}
    length: 2
  ▶ [[Prototype]]: Array(0)

```

```

const uall = async (req, res) => {

  try {

    console.log("Fetching all users...");

    const all = await user.find(); // Add a breakpoint or log here

    console.log("Fetched users:", all);

    return res.status(200).json(all);

  } catch (error) {

    console.error("Error in uall:", error);

    return res.status(400).send("Server error...");

  }

};

```

```

40 const uall=async(req,res)=>{
41   try{
42     console.log("datafetching...")
43     const all=await new Promise((resolve,reject)=>{
44       setTimeout(()=>{
45         resolve([
46           {
47             "id": 1,
48             "name": "Alice Johnson",
49             "email": "alice.johnson@example.com",
50             "age": 29,
51             "location": "New York",
52             "skills": ["JavaScript", "React", "Node.js"]
53           },
54           {
55             "id": 2,
56             "name": "Bob Smith",
57             "email": "bob.smith@example.com",
58             "age": 34,
59             "location": "San Francisco",
60             "skills": ["Python", "Django", "Machine Learning"]
61           }
62         ]),1000)
63       })
64     }
65     return res.status(200).json(all)
66   }
67   catch(error){
68     console.log("error found",error)
69     return res.status(400).send("json server");
70   }

```

```

71   }
72 }
73 const express = require('express');
74 const app = express();
75 app.get('/users', uall);
76 app.listen(3000, () => console.log('Server running on port 3000'));

```

```

77  const express = require('express');
78  const app = express();
79
80  app.use(express.json()); // Middleware to parse JSON request bodies
81
82  let records = [
83    {
84      id: 1,
85      name: "Alice Johnson",
86      email: "alice.johnson@example.com",
87      age: 29,
88      location: "New York",
89      skills: ["JavaScript", "React", "Node.js"],
90    },
91    {
92      id: 2,
93      name: "Bob Smith",
94      email: "bob.smith@example.com",
95      age: 34,
96      location: "San Francisco",
97      skills: ["Python", "Django", "Machine Learning"],
98    },
99  ];
100
101  const deleteUser = async (req, res) => {
102    try {
103      const userId = req.body.id;
104

```

```

123    }
124  };
125
126  // DELETE endpoint
127  app.delete('/delete', deleteUser);
128
129  app.listen(3000, () => console.log('Server running on port 3000'));
130

```

```

const deleteUser = async (req, res) => {

  try {

    const userId = req.body.id;

    if (!userId) {

      return res.status(400).send("User ID is required.");

    }


```

```
// Find and delete the user

const result = await User.findOneAndDelete({ id: userId });

if (!result) {

  return res.status(404).send("User not found.");

}

console.log(`User with ID ${userId} deleted.`);

return res.status(200).json({ message: "User deleted successfully", user: result });

} catch (error) {

  console.error("Error found:", error);

  return res.status(500).send("Server error.");

}

};

// DELETE Route

app.delete('/delete', deleteUser);

app.listen(3000, () => console.log('Server running on port 3000'));
```

```

135 // Middleware to parse JSON request bodies
136 app.use(express.json());
137
138 // Sample in-memory array to store data
139 let records = [
140   {
141     id: 1,
142     name: "Alice Johnson",
143     email: "alice.johnson@example.com",
144     age: 29,
145     location: "New York",
146     skills: ["JavaScript", "React", "Node.js"],
147   },
148   {
149     id: 2,
150     name: "Bob Smith",
151     email: "bob.smith@example.com",
152     age: 34,
153     location: "San Francisco",
154     skills: ["Python", "Django", "Machine Learning"],
155   }
156 ]
157
158
159 const adduser = async (req, res) => {
160   try {
161     const newuser = req.body;
162

```

```

162
163     // Validate required fields
164     if (!newuser.id || !newuser.name || !newuser.email) {
165       return res.status(400).send("id, name, and email fields are required");
166     }
167
168     // Check if the user already exists
169     const exist = records.find((record) => record.id === newuser.id);
170     if (exist) {
171       return res.status(400).send("User already exists");
172     }
173
174     // Add new user to records
175     records.push(newuser);
176
177     console.log("New user added:", newuser);
178     return res.status(201).json({ message: "User added successfully", user: newuser });
179   } catch (error) {
180     console.log("Error found:", error);
181     return res.status(400).send("Error occurred while adding the user");
182   }
183 };
184
185
186 app.post('/add', adduser);
187
188 app.listen(3000, () => console.log('Server running on port 3000'));
189

```

```
const adduser = async (req, res) => {  
  try {  
    const newUser = req.body;  
  
    // Validate required fields  
    if (!newUser.id || !newUser.name || !newUser.email) {  
      return res.status(400).send('id, name, and email fields are required');  
    }  
  
    // Check if the user already exists  
    const exist = await User.findOne({ id: newUser.id });  
    if (exist) {  
      return res.status(400).send('User already exists');  
    }  
  
    // Save new user to the database  
    const user = new User(newUser);  
    await user.save();  
  
    console.log('New user added:', newUser);  
    return res.status(201).json({ message: 'User added successfully', user: newUser });  
  } catch (error) {  
    console.log('Error found:', error);  
    return res.status(500).send('Error occurred while adding the user');  
  }  
};
```

```

192 const express = require('express');
193 const app = express();
194
195 let records = [
196   {
197     id: 1,
198     name: "Alice Johnson",
199     email: "alice.johnson@example.com",
200     age: 29,
201     location: "New York",
202     skills: ["JavaScript", "React", "Node.js"],
203   },
204   {
205     id: 2,
206     name: "Bob Smith",
207     email: "bob.smith@example.com",
208     age: 34,
209     location: "San Francisco",
210     skills: ["Python", "Django", "Machine Learning"],
211   },
212 ];
213
214 // Middleware to parse JSON requests
215 app.use(express.json());
216
217 // Function to update user
218 const updateUser = async (req, res) => {

```

```

219   try {
220     const { id } = req.body; // User ID to identify the record
221     const updatedData = req.body; // New data to update
222
223     // Validate ID is provided
224     if (!id) {
225       return res.status(400).send("User ID is required");
226     }
227
228     // Find the user index in the records
229     const index = records.findIndex((record) => record.id === id);
230
231     if (index === -1) {
232       return res.status(404).send("User not found");
233     }
234
235     // Update the user record
236     records[index] = { ...records[index], ...updatedData };
237
238     console.log("User updated:", records[index]);
239     return res.status(200).json({ message: "User updated successfully", user: records[index] });
240   } catch (error) {
241     console.log("Error found:", error);
242     return res.status(500).send("Server error occurred");
243   }
244 };
245
246 // PUT route to update user

```



```
const updatedUser = await User.findOneAndUpdate(
  { id }, // Filter to find the user
  updateFields, // Fields to update
  { new: true } // Return the updated user
);

if (!updatedUser) {
  return res.status(404).send("User not found.");
}

return res.status(200).json({
  message: "User updated successfully.",
  user: updatedUser,
});
} catch (error) {
  console.error("Error updating user:", error);
  return res.status(500).send("Server error.");
}
};
```

```
1  [
2      {
3          "id": 1,
4          "name": "Alice Johnson",
5          "email": "alice.johnson@example.com",
6          "age": 29,
7          "location": "New York",
8          "skills": [
9              "JavaScript",
10             "React",
11             "Node.js"
12         ]
13     },
14     {
15         "id": 2,
16         "name": "Bob Smith",
17         "email": "bob.smith@example.com",
18         "age": 34,
19         "location": "San Francisco",
20         "skills": [
21             "Python",
22             "Django",
23             "Machine Learning"
24         ]
25     }
26 ]
```