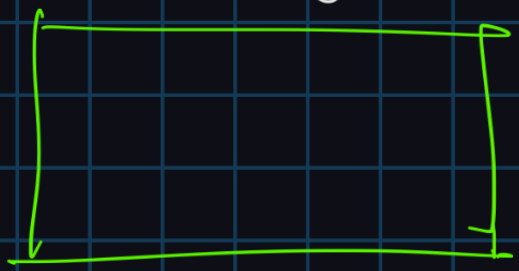
06-Install Node-Express And Use Fetch To Load Data

tip: Together to install both express and cors at the same time:

```
npm i express cors
```

Server

user management server



① package.json

```
"scripts": {
    "start": "node index.js"
}
```

② npm i express cors

③ .gitignore ⟶ node_modules

④ express default imports and layout

- `const express = require('express')`
- `const app = express()`
- `const port = process.env.PORT || 3000`

why this ?

- this has nothing to do with local server.
when website is deployed or an environment variable is used if is not possible to cater 500 users using only one port. So the server itself uses or delegates other ports to cater users and not only sticks to a fixed port.

⑤

```
app.get("/", (req, res) => {
    res.send("User Management Server is running")
})
```

⑥ app.listen(port, () => {
console.log("app is running succesfully at ${port}")})

---

⑦ let's create some hard coded data

```
let users = [
  { id:1 , name:"   " , email:"   " },
  { id:2 ,  "   "       "   "       },
  { id:3 ,  "   "       "   "       }
]
```

⑧ new API :

```
app.get("/users", (req, res) => {
    res.send(users)
})
```

CLIENT SIDE – users management client

① Setup vite + react router dom

② loading data and setting in state directly in main.jsx (for shortcut purpose)

```
const [users, setUsers] = useState([]);

useEffect (() => {
    fetch ("localhost:3000/users")
    .then ( res => res.json() )
    .then (data => setUsers (data))
}, [])
```

③ But data will not load as the server has not given permission to use them.

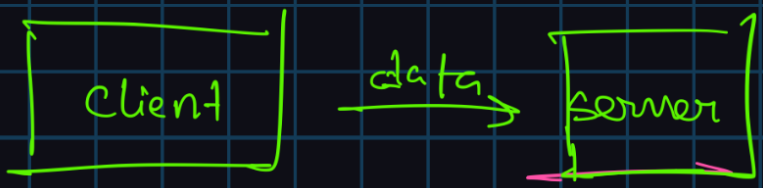
Server side: →

```
const cors = require ("cors")
app.use(cors())
```

④ Now you have got the data.
Show it on UI however you like.

---

now let us send data :



② const handleAddUser = event => {
event.preventDefault();
const form = event.target
const name = form.name.value
const email = form.email.value
const user = { name, email } }

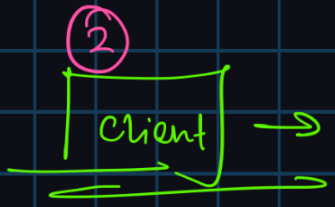
① <form onSubmit={handleAddUser}>
<input type="text" name="name" />
<input type="email" name="email" />
<input type="submit" value="Add User" />
</form>

Now we have to send this 'user' to the server using POST request.

1. Create a POST API on the server.

```
app.post('/users', (req, res) {
    console.log("post request hitting");
    console.log(req.body) } )
```

as the request will be coming from the body




```
const handleAddUser = event => {
        event.preventDefault();
    const form = event.target
    const name = form.name.value
    const email = form.email.value
    const user = { name, email }
```

here until now we did not give other parameters. we only fetched data normally.

(an object)

```
fetch ("localhost/users", {
    method : "POST",
    headers : {
        'content-type' : 'application/json'
            },
    body : JSON.stringify (user) }
  .then (res => res.json())
  .then (data => console.log ('inside post response', data))
```

(i) But at this point if we see the console of the server after clicking Submit, we will see:

→ "post request hitting from server sid" undefined ←

To solve this we need to use a middle ware in server side.

```
app.use(express.json());
```

② Now going to the post API :

```
app.post('/users', (req, res) {
    console.log("post request hitting");
    console.log(req.body)
    const newUser = req.body
    newUsed.id = users.length + 1
    users.push(newUser)
    res.send(newUser) });
```

as we are not using database yet we are experimenting like this to send it to the client side

③ Now we are going to show the user in the UI using Bangla System

[usually it will be automatically update through API but will learn it later]

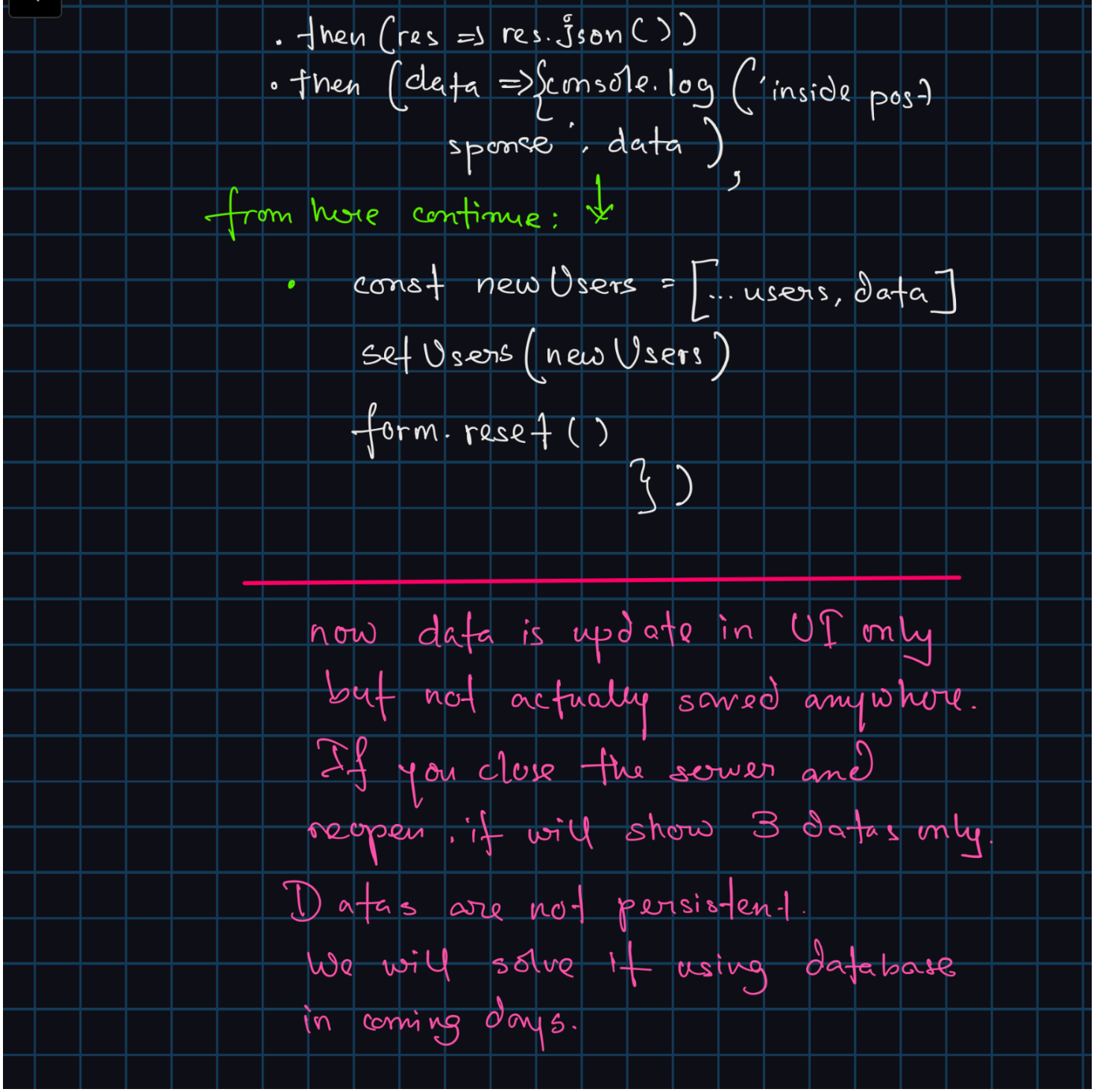
Back to this handler

```
const handleAddUser = event => {
    event.preventDefault();
const form = event.target
const name = form.name.value
const email = form.email.value
const user = { name, email }
```

here until now we did not give other parameters. we only fetched data normally.

(an object)

```
fetch("localhost/users", {
    method: "POST",
    headers: {
        'content-type': 'application/json'
    },
    body: JSON.stringify(user) }
```



Server Side:

index.js

```
const express = require("express");
const cors = require("cors");
const app = express();
const port = process.env.PORT || 3000;

// middleware:
app.use(cors());
app.use(express.json());

const users = [
   { id: 1, name: "Tushar", email: "tushar@tushar.com" },
   { id: 2, name: "Naruto", email: "naruto@naruto.com" },
   { id: 3, name: "Messi", email: "messi@messi.com" },
];
```

```
app.get("/", (req, res) => {
   res.send("Server is running successfully");
});

app.get("/users", (req, res) => {
   res.send(users);
});

app.post("/users", (req, res) => {
   console.log("post request hitting from server side");
   console.log(req.body);
   const newUser = req.body;
   newUser.id = users.length + 1;
   users.push(newUser);
   res.send(newUser);
});

app.listen(port, () => {
   console.log(`app running at port: ${port}`);
});
```

Client Side:
App.jsx:

```
import { useEffect, useState } from "react";

import "./App.css";

function App() {
   const [users, setUsers] = useState([]);
   useEffect(() => {
      fetch("http://localhost:3000/users")
         .then((res) => res.json())
         .then((data) => setUsers(data));
   }, []);

   const handleAddUser = (event) => {
      event.preventDefault();
      const form = event.target;
      const name = form.name.value;
      const email = form.email.value;

      const user = { name, email };
      console.log("normally loggin user: ", user);

      fetch("http://localhost:3000/users", {
         method: "POST",
         headers: {
            "content-type": "application/json",
         },
         body: JSON.stringify(user),
      })
         .then((res) => res.json())
```

```
            .then((data) => {
               console.log(data);
               const newUsers = [...users, data];
               setUsers(newUsers);
               form.reset();
            })
            .catch((err) => console.log(err));
      };
      return (
         <>
            <h1>User Management System</h1>
            <form onSubmit={handleAddUser}>
               <input type="text" name="name" placeholder="name" />
               <br />
               <input type="email" name="email" placeholder="email" />
               <br />
               <input type="submit" value="Add User" />
            </form>

            <h3>Number of users: {users.length}</h3>

            {users.map((user) => {
               return (
                  <p key={user.id}>
                     {user.id} : {user.name} : {user.email}
                  </p>
               );
            })}
         </>
      );
}

export default App;
```