CS351 – Lab 8.2: From HTML Forms to REST/JSON APIs

Purpose

This exercise extends Lab 8.1 (FormData). Students keep the existing Express application and compare two communication models: (1) traditional HTML form submission and (2) JavaScript/ REST communication using fetch().

Learning Objectives

• Understand HTML form processing versus REST APIs.
• Create an Express endpoint that returns JSON.
• Use JavaScript fetch() to call the backend.
• Update the page without a reload.
• Observe that the business logic changes very little.

Part A – Existing Traditional Application

Students begin with the existing HTML form:

```
<form action="/readCustomerInfoAndRespond" method="POST">
    <input name="name">
    <input name="email">
    <button type="submit">Submit</button>
</form>
```

Existing Express route:

```
router.post('/readCustomerInfoAndRespond', function(req,res){
    const name = req.body.name;
    const email = req.body.email;

    res.send("Welcome " + name +
             "<br>We will contact you at " + email);
});
```

Architecture

Browser → HTML Form → Express → HTML Response

Part B – Modern REST Version

Create a second endpoint that returns JSON instead of HTML.

```
router.post('/api/customer', function(req,res){
    const name = req.body.name;
    const email = req.body.email;

    res.json({
        message: "Welcome " + name,
        email: email
    });
});
```

Replace form submission with JavaScript fetch():

```
document.getElementById("submitBtn").addEventListener("click", async () => {

  const response = await fetch("/api/customer",{
      method:"POST",
      headers:{
          "Content-Type":"application/json"
      },
      body: JSON.stringify({
```

```
            name: document.getElementById("name").value,
            email: document.getElementById("email").value
        })
    });

    const data = await response.json();

    document.getElementById("result").innerHTML =
        data.message + "<br>" + data.email;
});
```

Architecture

Browser JavaScript / React → fetch() → Express REST API → JSON → Dynamic Page Update

Deliverables

1. Traditional HTML form still works.
2. New REST endpoint returns JSON.
3. JavaScript fetch() successfully calls the endpoint.
4. Page updates dynamically without a reload.
5. Student can explain the differences between the two architectures.

Discussion

IMPORTANT: both architectures are widely used today. Traditional server-rendered applications remain common, while SPAs typically communicate with REST APIs using fetch(). We will reuse this concept later in our Agentic Web example where the frontend communicates with an AI-enabled backend using exactly the same REST model.