# Implement Back-end in Profile Component

**Estimated duration**: 40 mins

Welcome to the **Implement Back end in Profile Component** lab. In this lab, you will implement functionality to edit user details. You are provided with the code for the user profile component. The initial render displays the user's name and email.

You will develop code which provides an option switch to `edit` mode, enabling the user to modify their name. The page also ensures user authentication, redirecting to the login page if the authentication token is missing. If the user is logged in, when the user submits the updated details, a PUT request should be made to the `/api/auth/update` endpoint, updating the user's profile on the server. Successful updates are then reflected in the session storage and the displayed profile.

## Objectives

- Use the `/api/auth/update` endpoint to handle user profile request to update user profile information.
- Integrate and handle authorization by including a Bearer token in the API request headers, enhancing security during profile update.
- Handle asynchronous API responses, interpreting and processing the JSON data returned from the profile endpoint.
- Enhance the client-side form handling by using a React state to manage user input for email and password.

## Prerequisite

Ensure that you have completed the previous tasks and pushed all the changes to GitHub. You should have already updated `giftlink-backend` and `giftlink-frontend` directories in your repository from the previous lab.

Clone the repository to this local environment that contains all the code that you pushed as a part of the previous lab before starting this lab, as these will be your working directories for the lab.

# Step 1: Implement API call for handleSubmit()

In this lab, you have to complete five tasks. The file that you cloned contains placeholders for each of the tasks listed below.

Call the API to implement the `handleSubmit()` functionality in the `Profile.js` file under the `giftlink-frontend` folder to set its method, headers, and body. Note that the `Profile.js` file contains the profile component code. You do not need to write the component; just implement the API call in it.

### Tasks

1. Set method
2. Set headers sections
3. Set body to send user details
4. Set the new name in the AppContext
5. Set user name in the session
6. Add the Profile component Route to `App.js`.

**Code Template**

```
const handleSubmit = async () => {
    try{
      const response = await fetch(`/api/auth/update`, {
           //Task 1: set method
           //Task 2: set headers
           //Task 3: set body to send user details
            if (response.ok) {
                       //Task 4: set the new name in the AppContext
                //insert code here
                //Task 5: set user name in the session
                //insert code here
                setUserDetails(updatedDetails);
                setEditMode(false);
                // Display success message to the user
                setChanged("Name Changed Successfully!");
                setTimeout(() => {
                setChanged("");
                navigate("/");
              }, 1000);
            } else {
              // Handle error case
              throw new Error("Failed to update profile");
            }
     })
      }catch (e) {
        console.log("Error updating details: " + e.message);
    }
}
```

## Hints

1. Hint to set post method

▼ Click here for a hint

Use `method: 'PUT'` to post user details

2. Hint to set headers section

▼ Click here for a hint

Use `headers: {...}` to set the headers

```
headers: {
        "Authorization": `Bearer ${authtoken}`,
        "Content-Type": "application/json",
        "Email": email,
      },
```

3. Hint to set body section to send user details

▼ Click here for a hint

Use `body: JSON.stringify` to set body variables to send user details to backend

```
body: JSON.stringify(payload),
```

4. Hint to set user name in AppContext

▼ Click here for a hint

Use `setUserName()` to set updated user name

```
setUserName(updatedDetails.name);
```

5. Hint to set user name in session

▼ Click here for a hint

Use `sessionStorage.setItem("<attr>","<val>")` to set updated user profile.

```
sessionStorage.setItem("name", updatedDetails.name);
```

6. Add the Profile component to `App.js`.

▼ Click here for a hint
- Add the following code to the list of imports.

```
import Profile from './components/Profile/Profile';
```

- Add the following code along with the other routes.

```
<Route path="/app/profile" element={<Profile/>} />
```

# Step 2: Push Changes to GitHub

After implementing and testing the API endpoints, push your changes to GitHub.

### Tasks

Task 1: Stage changes
Task 2: Commit changes
Task 3: Push changes

### Hints for Git Commands

1. Hint for staging changes

▼ Click here for a hint

Use `git add .` to stage all your changes for commit

2. Hint for committing changes

▼ Click here for a hint

Commit your staged changes with `git commit -m "Your message here"`

3. Hint for pushing to GitHub

▼ Click here for a hint

Finally, push your commits to GitHub with `git push`

# Evidence and Solution

## Solution

▼ Click here for sample completed code

```
import React, { useEffect, useState } from "react";
import { useNavigate } from "react-router-dom";
import './Profile.css'
import {urlConfig} from '../../config';
import { useAppContext } from '../../context/AuthContext';
const Profile = () => {
  const [userDetails, setUserDetails] = useState({});
 const [updatedDetails, setUpdatedDetails] = useState({});
 const {setUserName} = useAppContext();
 const [changed, setChanged] = useState("");
 const [editMode, setEditMode] = useState(false);
  const navigate = useNavigate();
  useEffect(() => {
    const authtoken = sessionStorage.getItem("auth-token");
    if (!authtoken) {
      navigate("/app/login");
    } else {
      fetchUserProfile();
    }
  }, [navigate]);
  const fetchUserProfile = async () => {
    try {
      const authtoken = sessionStorage.getItem("auth-token");
      const email = sessionStorage.getItem("email");
      const name=sessionStorage.getItem('name');
      if (name || authtoken) {
                const storedUserDetails = {
                  name: name,
                  email:email
                };
                setUserDetails(storedUserDetails);
                setUpdatedDetails(storedUserDetails);
              }
} catch (error) {
  console.error(error);
  // Handle error case
}
};
const handleEdit = () => {
setEditMode(true);
};
const handleInputChange = (e) => {
setUpdatedDetails({
  ...updatedDetails,
  [e.target.name]: e.target.value,
});
};
const handleSubmit = async (e) => {
  e.preventDefault();
  try {
    const authtoken = sessionStorage.getItem("auth-token");
    const email = sessionStorage.getItem("email");
    if (!authtoken || !email) {
      navigate("/app/login");
      return;
    }
```

```
    const payload = { ...updatedDetails };
    const response = await fetch(`${urlConfig.backendUrl}/api/auth/update`, {
      method: "PUT",//Step 1: Task 1
      headers: {//Step 1: Task 2
        "Authorization": `Bearer ${authtoken}`,
        "Content-Type": "application/json",
        "Email": email,
      },
      body: JSON.stringify(payload),//Step 1: Task 3
    });
    if (response.ok) {
      // Update the user details in session storage
      setUserName(updatedDetails.name);//Step 1: Task 4
      sessionStorage.setItem("name", updatedDetails.name);//Step 1: Task 5
      setUserDetails(updatedDetails);
      setEditMode(false);
      // Display success message to the user
      setChanged("Name Changed Successfully!");
      setTimeout(() => {
        setChanged("");
        navigate("/");
      }, 1000);
    } else {
      // Handle error case
      throw new Error("Failed to update profile");
    }
  } catch (error) {
    console.error(error);
    // Handle error case
  }
};
return (
<div className="profile-container">
  {editMode ? (
<form onSubmit={handleSubmit}>
<label>
  Email
  <input
    type="email"
    name="email"
    value={userDetails.email}
    disabled // Disable the email field
  />
</label>
<label>
   Name
   <input
     type="text"
     name="name"
     value={updatedDetails.name}
     onChange={handleInputChange}
   />
</label>
<button type="submit">Save</button>
</form>
) : (
```

```
<div className="profile-details">
<h1>Hi, {userDetails.name}</h1>
<p> <b>Email:</b> {userDetails.email}</p>
<button onClick={handleEdit}>Edit</button>
<span style={{color:'green',height:'.5cm',display:'block',fontStyle:'italic',fontSize:'12px'}}>{changed}</span>
</div>
)}
</div>
);
};
export default Profile;
```

At this point you should test the front-end back-end integration to ensure it is all working as expected at this stage.

## Evidence

Make note of the GitHub URL for your repository. You will be asked to submit this for peer evaluation. Ensure you completed all tasks in each step.

## Author(s)

Skills Network