ENSF 381
Full Stack Web Development

Lecture 30:
Examples in Authentication and Data Management

Ahmad Abdellatif, PhD

UNIVERSITY OF CALGARY

Web authentication example

- We will create a username/password authentication system.
- The backend manages a list of users.
- The frontend includes a form for users to input their credentials. When submitted, the frontend sends the information to the backend for authentication.
- Success or failure messages are then displayed.

Web authentication example - Frontend

```
import React, { useState } from 'react';

function AuthenticationForm() {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const [message, setMessage] = useState('');

  function handleAuthentication() {
      fetch('http://127.0.0.1:5000/authenticate', {
          method: 'POST',
          headers: {
              'Content-Type': 'application/json',
          },
          body: JSON.stringify({'username':username, 'password':password}),
      })
      .then(response => {
          if (response.ok) {
              return response.json();
          } else {
              throw new Error('Authentication failed');
          }
      })
      .then(data => setMessage(data.message))
      .catch(error => setMessage('Authentication failed. Incorrect username or password.'));};
```

Web authentication example - Frontend

```
return (
      <div>
          <label>
              Username:
          </label>
              <input type="text" onChange={(e) => setUsername(e.target.value)} />
          <br />
          <label>
              Password:
          </label>
              <input type="password" onChange={(e) => setPassword(e.target.value)} />
          <br />
          <button onClick={handleAuthentication}>Submit</button>
          <br />
          <p>{message}</p>
      </div>
  );
};

export default AuthenticationForm;
```

Web authentication example – Backend

```python
from flask import Flask, jsonify, request
from flask_cors import CORS

app = Flask(__name__)
CORS(app)

# Array of user objects with username and password
users = [
  {"id": 1, "username": "user1", "password": "pass1"},
  {"id": 2, "username": "user2", "password": "pass2"},
]

# Route to authenticate user
@app.route('/authenticate', methods=['POST'])
def authenticate_user():
  data = request.get_json()
  entered_username = data.get('username')
  entered_password = data.get('password')

  # Check if the entered username and password match any user in the array
  for user in users:
      if user['username'] == entered_username and user['password'] == entered_password:
          return jsonify({"message": "Authentication successful"})

  return jsonify({"message": "Authentication failed. Incorrect username or password."})

if __name__ == '__main__':
  app.run()
```

Web authentication example

Redirecting to Dashboard on Correct Username and Password

Web authentication example - Backend

```python
from flask import Flask, jsonify, request
from flask_cors import CORS

app = Flask(__name__)
CORS(app)

# Array of user objects with username and password
users = [
  {"id": 1, "username": "user1", "password": "pass1"},
  {"id": 2, "username": "user2", "password": "pass2"},
]

# Route to authenticate user
@app.route('/authenticate', methods=['POST'])
def authenticate_user():
  data = request.get_json()
  entered_username = data.get('username')
  entered_password = data.get('password')

  # Check if the entered username and password match any user in the array
  for user in users:
      if user['username'] == entered_username and user['password'] == entered_password:
          return jsonify({"authenticated": True, "message": "Authentication successful"})

  return jsonify({"authenticated": False, "message": "Authentication failed. Incorrect username or password."})

if __name__ == '__main__':
  app.run()
```

Web authentication example – Frontend (App.js)

```
import React from 'react';
import { BrowserRouter, Route, Routes } from 'react-router-dom';
import AuthenticationForm from './AuthenticationForm';
import Dashboard from './dashboard';

function App() {
  return (
      <BrowserRouter>
          <Routes>
              <Route path="/dashboard" element={<Dashboard />} />
              <Route path="/" element={<AuthenticationForm />} />
          </Routes>
      </BrowserRouter>
  );
};

export default App;
```

Web authentication example – Frontend (Dashboard.js)

```
import React from 'react';

function Dashboard() {
  return (
      <div>
          <h2>Welcome to the Dashboard!</h2>
      </div>
  );
};

export default Dashboard;
```

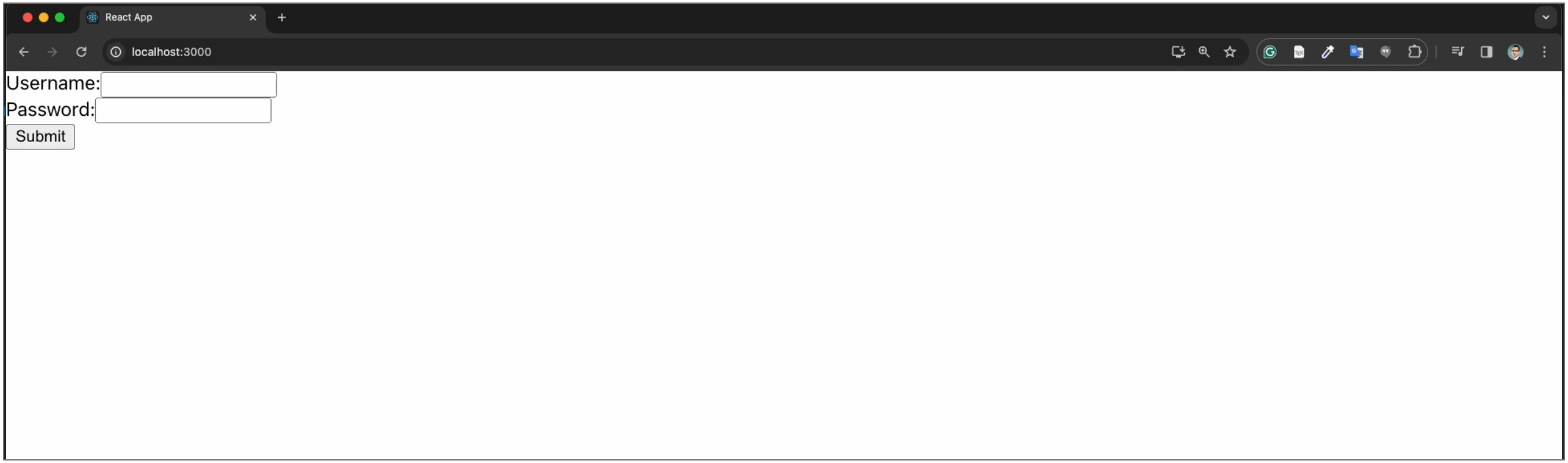
Web authentication example – Frontend (AuthenticationForm.js)

```
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';

function AuthenticationForm() {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const [message, setMessage] = useState('');
  const [authenticated, setAuthenticated] = useState(false);
  const navigate = useNavigate();

  function handleAuthentication() {
      fetch('http://127.0.0.1:5000/authenticate', {
          method: 'POST',
          headers: {
              'Content-Type': 'application/json',
          },
          body: JSON.stringify({'username':username, 'password':password}),
      })
      .then(response => response.json())
      .then(response => {

          if (response.authenticated) {
              setAuthenticated(true);
              setMessage("Authentication successful");
          } else {
              setAuthenticated(false);
              setMessage("Authentication failed. Incorrect username or password.");
          }
      })
      .catch(error => setMessage('Authentication failed. Incorrect username or password.'));
  };

  if (authenticated) {
      // Redirect to another page after successful authentication
      navigate("/dashboard")
  }
```

Web authentication example – Frontend (AuthenticationForm.js)

```
return (
      <div>
          <label>
              Username:
              <input type="text" onChange={(e) => setUsername(e.target.value)} />
          </label>
          <br />
          <label>
              Password:
              <input type="password" onChange={(e) => setPassword(e.target.value)} />
          </label>
          <br />
          <button onClick={handleAuthentication}>Submit</button>
          <br />
          <p>{message}</p>
      </div>
  );
};

export default AuthenticationForm;
```

Web authentication example



Dynamic Person Object Management

- The project involves building a Full Stack application that allows users to perform CRUD (Create, Read, Update, Delete) operations on a collection of person objects.
- The backend provides API endpoints to manage the array of person objects.
- The frontend allows users to interact with the backend through HTTP requests for adding, viewing, updating, and deleting persons.
- The application features a simple user interface with buttons to add new persons, update existing ones, and delete them.

Dynamic Person Object Management - Backend

```python
from flask import Flask, jsonify, request
from flask_cors import CORS


app = Flask(__name__)
CORS(app)

# Array of person objects
persons = [
  {"id": 1, "name": "John Doe", "age": 25},
  {"id": 2, "name": "Jane Smith", "age": 30},
]

# Routes
@app.route('/persons', methods=['GET'])
def get_persons():
  return jsonify(persons)

@app.route('/persons', methods=['POST'])
def add_person():
  new_person = request.get_json()
  new_person['id'] = len(persons) + 1
  persons.append(new_person)
  return jsonify(new_person) # A function that converts a Python dictionary into a JSON response object.

@app.route('/persons/<person_id>', methods=['PUT'])
def update_person(person_id):
  for person in persons:
      if person['id'] == person_id:
          updated_person = request.get_json()
          person.update(updated_person)
          return jsonify(person)
  return jsonify({"error": "Person not found"})

@app.route('/persons/<person_id>', methods=['DELETE'])
def delete_person(person_id):
  global persons
  persons = [person for person in persons if person['id'] != person_id]
  return jsonify({"message": "Person deleted successfully"})

if __name__ == '__main__':
  app.run()
```

Dynamic Person Object Management - Frontend

```
import React, { useState, useEffect } from 'react';

function PersonList() {
  // State to hold the list of persons
  const [persons, setPersons] = useState([]);

  // Fetch persons from the server on component mount
  useEffect(() => {
      // Fetch data from the server
      fetch('http://127.0.0.1:5000/persons')
          .then(response => response.json()) // Parse the response as JSON
          .then(data => setPersons(data)) // Set the persons state with the fetched data
          .catch(error => console.error('Error fetching persons:', error));
  }, []);

  // Function to add a new person to the server
  function handleAddPerson() {
      // Define a new person object
      const newPerson = { name: 'New Person', age: 25 };

      // Send a POST request to add the new person to the server
      fetch('http://127.0.0.1:5000/persons', {
          method: 'POST',
          headers: {
              'Content-Type': 'application/json', // Specify the content type as JSON
          },
          body: JSON.stringify(newPerson), // Convert the new person object to JSON
      })
      .then(response => response.json()) // Parse the response as JSON
      .then(data => setPersons([...persons, data])) // Update the state with the new person
      .catch(error => console.error('Error adding person:', error)); };// Log any errors
```

Dynamic Person Object Management - Frontend

```javascript
// Function to update an existing person on the server
function handleUpdatePerson(id) {
    // Define an updated person object
    const updatedPerson = { name: 'Updated Person', age: 30 };

    // Send a PUT request to update the person on the server
    fetch(`http://127.0.0.1:5000/persons/${id}`, {
        method: 'PUT',
        headers: {
            'Content-Type': 'application/json', // Specify the content type as JSON
        },
        body: JSON.stringify(updatedPerson), // Convert the updated person object to JSON
    })
    .then(response => response.json()) // Parse the response as JSON
    .then(data => {
        // Update the state with the updated person
        setPersons(persons.map(person => (person.id === id ? data : person)));
    })
    .catch(error => console.error('Error updating person:', error));
};
```

```javascript
// Function to delete a person from the server
function handleDeletePerson(id) {
    // Send a DELETE request to remove the person from the server
    fetch(`http://127.0.0.1:5000/persons/${id}`, {
        method: 'DELETE',
    })
    .then(() => setPersons(persons.filter(person => person.id !== id))) // Update the state by removing the deleted person
    .catch(error => console.error('Error deleting person:', error));
};
```

Dynamic Person Object Management – Frontend

```
  return (
      <div>
          <button onClick={handleAddPerson}>Add Person</button>

          <ul>
              {persons.map(person => (
                  <li>
                      {person.name} - {person.age}

                      <button onClick={() => handleUpdatePerson(person.id)}>Update</button>

                      <button onClick={() => handleDeletePerson(person.id)}>Delete</button>
                  </li>
              ))}
          </ul>
      </div>
  );
};

export default PersonList;
```

Dynamic Person Object Management

Questions