ENSF-381: Full Stack Web Development Laboratory

Ahmad Abdellatif and Novarun Deb
Department of Electrical & Software Engineering
University of Calgary
Lab 6

Objectives

Welcome to the ENSF381 course lab! In this lab, we will focus on fundamental React concepts. Specifically, we will explore the fundamentals of React by practicing JSX, creating reusable components, and passing data using props. we will also learn how to render lists efficiently and improve code organization by modularizing components into separate files.

Groups

Lab instructions must be followed in groups **of two students**.

Submission

You must submit the complete source code, ensuring it can be executed without any modifications. Also, if requested by the instructor, you may need to submit the corresponding documentation file (e.g., word and image). Only one member of the group needs to submit the assignment, but the submission must include the names and UCIDs of all group members at the top of the code.

Deadline

Lab exercises must be submitted by **11:55 PM on the same day as the lab session**. Submissions made within 24 hours after the deadline will receive a maximum of 50% of the mark. Submissions made beyond 24 hours will not be evaluated and will receive a grade of zero.

Academic Misconduct

Academic Misconduct refers to student behavior which compromises proper assessment of a student's academic activities and includes: cheating; fabrication; falsification; plagiarism; unauthorized assistance; failure to comply with an instructor's expectations regarding conduct required of students completing academic assessments in their courses; and failure to comply with exam regulations applied by the Registrar.

For more information on the University of Calgary Student Academic Misconduct Policy and Procedure and the SSE Academic Misconduct Operating Standard, please visit: https://schulich.ucalgary.ca/current-students/undergraduate/student-resources/policies-and-procedures

Exercise 1: Understanding JSX Syntax in React

Objective: In this exercise, you will practice using JSX, a syntax extension for JavaScript that allows you to write HTML-like code inside JavaScript. You will create a basic React component and experiment with embedding JavaScript inside JSX.

1. **Create a React App**
   - Open your terminal and create a new React app using the command:
     ```
     npx create-react-app jsx-exercise
     ```
   - Navigate to the jsx-exercise folder and open the project in your IDE.

2. **Create Your First JSX Element**
   - In the App component (`src/App.js`), replace the default code with a simple JSX element.
   - The element should include a heading (<h1>) that says "ENSF-381: Full Stack Web Development"
   - Below the heading, include a paragraph (<p>) with the text "React Components".

3. **Embed JavaScript Expressions in JSX**
   - Inside the App component, create a variable `currentYear` that holds the current year. Hint: use JavaScript's Date() method.
   - Create a new <p> element that displays `currentYear` inside the JSX by embedding the variable within curly braces {}.

4. **Using a Conditional Statement in JSX**
   - Create a boolean variable `isLoggedIn` and set it either to true or false.
   - Inside the JSX, conditionally render a message using a ternary operator. If isLoggedIn is true, display "Welcome back!" otherwise, show "Please log in.".

Submission:

1- Fill out the names and UCIDs of all group members in Answer_sheet.
2- Create a new GitHub repository and upload the code for Exercise 1 to the new repository. **NOTE: Do not include the node_modules folder.**

Exercise 2: Creating Components in React

Objective:
In this exercise, you will create and use React components. You will build a simple, reusable component and learn how to pass data (via props) between components.

1. **Create Three Components**
    - Using the same React App you created in the previous exercise, create three new files: `Home.js`, `About.js`, and `Contact.js` inside the `src` folder.
    - Each file should contain a new React component that returns a simple message indicating the section (e.g., "Welcome to the Home Page", "About Us", "Contact Us").
2. **Use the Components in App.js**
    - In `src/App.js`, import the `Home`, `About`, and `Contact` components.
    - Inside the `App` component, display all three components.
3. **Pass Props to Components**
    - Modify each component to accept a title prop and display it as a heading.
    - When using these components in App.js, pass appropriate titles like:
        - Home Page
        - About Us
        - Contact Us
    - Modify each component to accept a description prop and display a short paragraph (<p>) to display in the page (component).

    - Update `App.js` to pass different descriptions to each component:

        - **Home Page:** Display the description "Welcome to our website."
        - **About Page:** Display the description "We are passionate about delivering quality experiences."
        - **Contact Page**: Display the description "Feel free to reach out to us via email or phone."

Submission:
Upload the code for Exercise 2 to the same repository you created for Exercise 1. **NOTE: Do not include the node_modules folder.**

Exercise 3: Working with Lists and Modularization in React

Objective:
In this exercise, you will practice rendering lists in React and organizing your code by modularizing components into separate files. You will display a list of **important topics in different engineering disciplines**.

1. **Create a New Component for the Engineering Topics List**
    - Inside the `src` folder, create a new file called `EngineeringTopics.js`
    - Create a new React component named `EngineeringTopics` that displays a list of engineering topics. Add heading (<h2>) that shows "Engineering Topic".
    - Use the following array to store titles and descriptions of the engineering topics:

```
const topics = [
{title: "Software Engineering", description: "Building innovative software solutions for the modern world."},
{title: "Electrical Engineering", description: "Powering innovation in electronics and systems."},
{title: "Mechanical Engineering", description: "Designing machines and systems that shape the future."},
{title: "Chemical Engineering", description: "Advancing processes for a sustainable future."}
];
```

2. **Render the Topics Dynamically**
    - Inside the EngineeringTopics component, use `.map()` to dynamically render each engineering topic as follows:
        - <h3> element for the title.
        - <p> element for the description.

3. **Integrate the Component into the App component**
    - Import the EngineeringTopics component into `App.js`
    - Use the EngineeringTopics component inside the App component to display the list.

Submission:

1. Upload the code for Exercise 3 to the same repository you created for Exercise 1. **NOTE: Do not include the node_modules folder.**
2. Compress both Exercise 1, Exercise 2, and Exercise 3 into a single file and upload the compressed file to D2L. Also, upload the completed Answer_sheet.docx. **NOTE: Do not include the node_modules folder.**