ReactJS Part-5 - Lab Assignment

Lab Experiment Questions on React Dataflow and Styling

---

1. Building a Dynamic Contact List:

Create a **Contact List** application where:

- You can add, edit, and delete contacts.
- Use **state** to manage the list and **props** to pass contact details to child components.
- Apply styles using **CSS stylesheets**.

Hint:

- Use `useState` for the contact list.
- Pass contact data as props and validate using **prop-types**.

---

2. Product Catalog with Inline Styling:

Develop a **Product Catalog** that:

- Displays a list of products with **name, price, and availability**.
- Style each product card using **inline styles**.
- Change the background color based on availability.

Hint:

- Use **ternary operators** for dynamic inline styles.
- Use **props validation** to ensure the price is a number.

---

3. Theme Switcher using CSS Modules:

Build a **Theme Switcher** app:

- Allow toggling between **Light** and **Dark** modes.
- Use **CSS Modules** for scoped styles.
- Change font colors and backgrounds based on the theme.

Hint:

- Use state (`useState`) to track the theme.
- Dynamically apply CSS Module classes using `className={styles.className}`.

---

4. Styled Profile Card:

Create a **Profile Card** that displays user information like **name, age, and location**.

- Style using a combination of **inline styles**, **CSS stylesheets**, and **CSS Modules**.
- Ensure proper **props validation**.

Hint:

- Use a mix of styling approaches for learning versatility.
- Validate props like `age` as a number using **prop-types**.

---

5. Task Tracker with State and Props:

Create a **Task Tracker** app where:

- Users can add, mark complete, and delete tasks.
- Display the count of completed and pending tasks.
- Use **props validation** for each task object.
- Style the completed tasks with a **strikethrough** using **CSS stylesheets**.

Hint:

- Use `useState` for task management.
- Use `.map()` to render tasks and apply conditional styles.

---

6. Stylish Calculator:

Develop a **Calculator** that performs basic arithmetic operations.

- Style the calculator buttons using **inline styles**.
- Pass numbers and operators as **props**.
- Validate props to ensure only valid numbers and operators are passed.

Hint:

- Use `useState` to handle calculations.
- Use inline styles for button aesthetics.

---

7. Product Review System:

Create a **Product Review** component:

- Display **product name, image, description**, and user reviews.
- Use **CSS Modules** for styling the review section.
- Validate the props like rating (number between 1-5).

Hint:

- Manage reviews using `useState`.
- Ensure rating is validated correctly using **prop-types**.

---

8. Dynamic Form Styling:

Create a **Signup Form** with fields like **name, email, and password**.

- Style the form using a combination of **CSS stylesheets** and **inline styles**.
- Validate inputs using state and show errors in a **styled error message**.

Hint:

- Use controlled components (`useState`) for input handling.
- Apply conditional inline styles for error messages.

---

9. E-commerce Product Filter:

Build a **Product Filter** app:

- Display a list of products with categories like **Electronics, Clothing, Home Decor**.
- Filter products dynamically by category.
- Style the components using **CSS Modules**.

Hint:

- Use state (`useState`) for category filtering.
- Use **props** to pass filtered data and validate appropriately.

---

10. News Feed Dashboard:

Develop a **News Feed** application:

- Fetch and display news articles with **title, description, and author**.
- Style the feed using **CSS stylesheets** for overall layout.
- Highlight featured articles with **inline styles**.

Hint:

- Use `useEffect` to fetch data.
- Apply inline styles conditionally for featured news.
