Internship Program

Objective:

- Introduce the interns to new technologies (JavaScript, HTML5, CSS, Material UI, React Js, Node Js and Mongo DB).
- Develop interns' analytic and problem-solving skills.
- Help interns develop newly acquired skills using challenging tasks and very a compact program.
- Introduce the interns to layering architecture with its benefits and implementation.
- Use all acquired skills and knowledge to transform real requirements into a fully functional product.

Program:

- **Install VS Code and Node Js**
- First Phase:
  - Objective: Build a basic internship performance system with CRUD operations .

    - First Task: Create Header and Links .
      Objective: Design and implement a header component with navigation links using Material UI, focusing on conditional rendering, data flow with props, and ensuring a visually appealing design.

        - **Header Design:** Include a header with a logo on the left and navigation links on the right.
        - **Navigation Links:** Include links like "Home", "Login", "Users", etc.
        - **Conditional Rendering:** Use conditional rendering to display different content based on application state or user actions (e.g., show different content when navigating between sections or based on user interaction).
        - **Data Flow with Props:** Ensure components communicate using props:
          Child to Parent: Use props to send data from child components (like navigation links) to the parent component (homepage).
          Parent to Child: Pass props down from the parent component (homepage) to child components (like page content components).
        - **Design:** Ensure the homepage has a visually appealing design.

    - Second Task: Create User Creation Form and Display Users
      Objective: Build a form to create users with specified fields and display user information in cards using Material UI.
      - User Creation Form:
        - Create a form with fields for: ID ,First Name (input), Last Name (input), Email (input), Phone Number (input) , Company (input), Division (dropdown/select with options: "IT Support" and "Dev Support/IT"), Starting Date (date picker) .
        - Use Material UI components for form elements.
        - Implement form validation for required fields (first name, last name, email) and format validation (email format, phone number format).
        - Use a provider for the date picker component at the root of the project .
      - Display Users in Cards:
        - Below the form, display cards for each user added.
        - Each card should visually represent user information (e.g., name, email, phone number, company, division, starting date).
        - Use Material UI card components for displaying user information.
        - Implement mapping of user data to dynamically create cards as users are added via the form.

      - Functionality:
        - When the user submits the form, add the user's information to a list of users.
        - Update the UI to display a card for each user added, showing their details.
        - Ensure the form resets after submission or provides clear feedback upon successful submission.
      - Responsive Design and Styling:
        - Implement responsive design principles using Material UI's grid system or breakpoints to ensure optimal layout across different devices.
        - Test and adjust the layout for mobile (small screens), tablet, and desktop viewports.

- Third Task: User Editing with Card Actions
  Objective: Implement functionality to edit user details using icons or buttons on user cards, displayed in a Material-UI card format.
  - Card Actions for Edit:
    - Include an icon (e.g., edit icon from Material-UI icons library) on each card that, when clicked, opens a dialog for editing that specific user's details.
    - Distinguish between MVC controllers and API controllers.
  - Edit Dialog:
    - Implement a dialog using Material-UI components (`Dialog`, `Dialog Title`, `Dialog Content`, `Text Field`, `Button`, etc.) for editing user details.
    - Pre-fill the dialog fields with the selected user's default values when the edit icon is clicked.
    - Allow users to modify existing details..
  - Form Validation and Submission:
    - Ensure form validation for required fields (First Name, Last Name, Email) and format validation (Email format, Phone Number format) within the edit dialog.
    - Update the selected user's details in the list and reflect the changes in the user card.

- Fourth Task: User Management with React Router DOM and Redux
  Objective: Implement user management functionality using React Router DOM for navigation and Redux for state management.
  - Navigation with React Router DOM:
    - Configure React Router DOM to manage navigation between pages/components..
    - Include routes for : Home page , User creation page , User list (view users) page.
  - User Creation Page:
    - Implement a separate page/component for creating a new user.
    - On form submission, add the new user to the Redux state and redirect to the User List page.
  - User List Page:
    - Implement a separate page/component for viewing users.
    - Display a list of users using Material-UI cards.
    - Include an edit icon on each card that, when clicked, opens a dialog for editing that specific user's details
    - Include a delete button on each card to remove the user.
  - Redux Integration:
    - Implement a separate page/component for viewing users.
    - Define Redux actions and reducers to : Add new users , Edit existing users, Delete users.
    - Use Redux to store and manage user data across components/pages.

- Redux Structure:
    1. Actions :
        - `FETCH_USERS`: Fetch user data from an API and store it in the Redux state.
        - `ADD_USER`: Add a new user to the Redux state.
        - `EDIT_USER`: Edit an existing user in the Redux state.
        - `DELETE_USER`: Delete a user from the Redux state