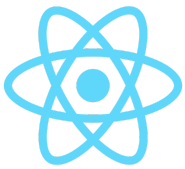
LabConnect Documentation Guide (updated March 2025)
Sagar Sahu, Rafael Cenzano, Will Broadwell

Application tech stack:

- Backend
  - PostgreSQL
  - Flask
  - Python
- Frontend
  - React.js
  - TailwindCSS
  - TypeScript



Tailwind CSS

Annotated Bibliography of Resources Used:

- Online resources:
  - https://www.youtube.com/watch?v=ydkQlJhodio&ab_channel=Fireship
    - This video taught me a few things about using TypeScript:
      - Benefits of TSX with React by enabling static type checking, which can lead to earlier detection of errors during development
  - https://www.youtube.com/watch?v=SqcY0GlETPk&ab_channel=ProgrammingwithMosh
    - This video helped review some of the skills I learned last semester for using React.js on the frontend, and established new skills like:
      - Understanding components, props, state, and event handling
      - Exploring the lifecycle methods of React components
      - Using hooks like useState and useEffect for managing state and side effects for user interactions
      - Implementing navigation using React Router
      - Managing global state with Context API and Redux

Getting Started Guide:

- To start running the frontend view:
  - Navigate to the root project directory: ‘cd ../frontend’
  - Run ‘npm start’
  - Run your localhost in a secure browser (e.g. Chrome, Safari)
- Basic features:
  - Home page
    - Welcome/main section
    - Project description section
    - Meet the team section
    - Contact/message section

- Jobs page
- Create research opportunity page
- Department staff page
- Profile page

What I learned so far:

- React & Next.js Concepts
  - Component Composition – I broke down sections into smaller, reusable components for better maintainability.
  - Props & State Management – I passed data between components using props and managed form inputs using useState.
  - Event Handling – I implemented button clicks, form submissions, and scroll events.
  - Refs (useRef) – I used references to scroll to specific sections dynamically.

- TypeScript Fundamentals
  - Typing State & Props – I made sure that useState and event handlers have proper TypeScript types (e.g., React.ChangeEvent<HTMLInputElement>).
  - Type Safety for Hooks – I helped prevent runtime errors by correctly typing useRef (useRef<HTMLDivElement>(null)).
  - Function Parameter & Return Types – I defined function signatures explicitly for better readability and debugging.

- JavaScript & DOM Manipulation
  - Scroll Event Listeners – I used window.addEventListener("scroll", handleScroll) to track page position.
  - Smooth Scrolling – I utilized scrollIntoView({ behavior: "smooth" }) for a better user experience.
  - Conditional Rendering – I modified the visibility of UI elements dynamically (showReturnToTop state for the return button).

- Styling & UI/UX Enhancements
  - TailwindCSS Utility Classes – I added responsive design efficiently with classes like hover:bg-blue-700, shadow-md, etc.
  - Responsive Layouts – I used flex-wrap, text-center, and w-11/12 md:w-3/4 to ensure elements adjust on different screen sizes.
  - Hover Effects & Animations – I improved UI interactivity with hover:shadow-xl transition-all duration-300.

- React Router
  - Navigation Between Pages – I used <Link to="/jobs"> to allow navigation without full-page reloads.

Updates made to the homepage of the frontend app so far: