University of Information Technology

Faculty of Information Systems

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Lab Report 3: Introduction to React

MSIS207: Web Application Development

Student name: Nguyễn Thiện Nhân – 23521084
Class: MSIS207.Q14.CTTT
Advanced Program in Information System – UIT

Contents

1 Student Information

Name: Nguyễn Thiện Nhân
Student ID: 23521084
Course: Web Development - MSIS207.Q14.CTTT

GitHub Repository: `github.com/nhanbayern/UIT_Web_Lab03_MSIS207.Q14.CTTT`
Public Demo URL: `uit-web-lab03-msis-207-q14-cttt.vercel.app`

2 Project Overview

The application represents **Lab 3: Introduction to React** for the course **Web Development – MSc. Tran Vinh Khiem**. It demonstrates the foundational principles of React, including component-based architecture, JSX, props, and state management, implemented using **React (Vite)** with modern ES modules.

The final demo integrates:

1. **UserProfile:** A component-based example demonstrating JSX rendering and prop passing.

2. **Counter:** A simple interactive component using `useState()` for state updates.

3. **Login:** A controlled form example using `onChange` and `useState()` to manage user input.

4. **Accordion:** Demonstrates lifting state up and parent-child communication.

5. **TodoApp (Kanban Board):** A fully functional to-do system with CRUD operations and responsive layout.

3 Technical Implementation

3.1 React Component Architecture

The system follows React's declarative, component-driven design pattern:

- Each UI feature is encapsulated in a functional component (e.g., `UserProfile`, `Counter`, `Login`).
- Data flows from parent to child through **props**, ensuring unidirectional data flow.
- State variables are managed locally within components using React's built-in `useState()` hook.

The entry point `main.jsx` renders the root component `<App />` to the DOM via Vite's modern bundler environment. All components are organized by functionality for maintainability and clarity.

3.2 State Management with Hooks

The project extensively uses the `useState()` hook to maintain component-level memory:

- **Counter:** Tracks numerical state that increments on each button click.
- **Login:** Controls two-way data binding for username and password fields.
- **TodoApp:** Manages a dynamic task array, allowing add, toggle, delete, and status transitions across workflow stages.

The state logic for `TodoApp` is enhanced by:

1. Centralized data management at the parent level.
2. Functions passed as props to children for event-driven updates.
3. Component re-rendering automatically triggered upon state mutation.

3.3 Kanban Board and Interactivity

The final enhancement transforms the simple to-do list into an interactive **Kanban board**:

- Tasks are organized into three columns: `Todo`, `In Progress`, and `Done`.

- Each task is rendered as a pastel-colored card with unique background shades generated randomly.
- Tasks support interactive actions:
  - `Move` – Transfers a task to the next column.
  - `Back` – Returns a task to the previous column.
  - `Done` – Marks a task as completed.
  - `Delete` – Removes a task permanently.
- Layout dynamically adjusts with CSS Flexbox and media queries for mobile and desktop responsiveness.

3.4 Styling and Theming

The visual design follows modern minimalism:

- Typography imported via Google Fonts using `Open Sans`:

```
    <link href="https://fonts.googleapis.com/css2?
      family=Open+Sans:wght@400;600&display=swap"
      rel="stylesheet" />
```

- Layout implemented with CSS3 Flexbox and shadow effects for card elevation.
- Responsive breakpoints:
  - Desktop (1024px): Three columns displayed horizontally.
  - Tablet (768–1024px): Columns scaled and centered.
  - Mobile (768px): Columns stacked vertically.
- Each pastel color is randomly assigned from a light color palette for task distinction.

3.5 Debugging and Validation

Debugging and validation were performed using the **React Developer Tools** browser extension:

- Inspected component hierarchies and props in real time.
- Tracked hook state changes within `Counter` and `TodoApp`.
- Enabled highlight re-render visualization to optimize component updates.

Manual testing validated:

1. Correct form submission and controlled input updates.
2. Smooth task state transitions and consistent card color mapping.
3. Proper layout behavior across various device screen sizes.

4 Deployment

The project is deployed publicly using **Vercel**, configured with zero manual setup. Automatic CI/CD integration ensures every new push to the `main` branch triggers an instant build and redeploy to the live demo.

- GitHub Repository: `github.com/nhanbayern/UIT_Web_Lab03_MSIS207.Q14.CTTT`
- Vercel Live Demo: `uit-web-lab03-msis-207-q14-cttt.vercel.app/`

Deployment commands:

```
npm run build
vercel deploy --prod
```

The build output generated by Vite (`/dist`) is automatically served through Vercel's static hosting layer.

5 Conclusion

This project successfully demonstrates the key fundamentals of React development:

- Functional components and declarative UI design.
- State management using hooks (`useState()`).
- Controlled inputs and two-way data flow with props.
- Reusable composition patterns (Card, Accordion, Kanban Board).

The final Kanban-based To-Do application integrates all learned concepts into a cohesive, maintainable, and responsive web system that adheres to modern React development standards.