#56

How to change React Settings so users can use their needed accessibility settings

Purpose

The goal of this research was to understand how to configure React applications so users can enable accessibility settings such as screen reader compatibility, focus outlines, and contrast modes to meet WCAG 2.1 standards.

Key Findings

- React supports accessibility through ARIA attributes and semantic HTML.
- The React-ARIA and React-Axe libraries can automatically test and enforce accessibility standards.
- You can enable high contrast and font scaling using React Context or user preferences stored in state.
- Keyboard navigation can be improved using tabIndex and focus management utilities.
- React apps should avoid removing native focus outlines unless replaced with visible custom focus styling.

Implementation Notes

Example: Add Accessibility Support in React

```
// Example component with ARIA and keyboard support

function AccessibleButton({ label, onClick }) {

 return (

  <button

   aria-label={label}

   onClick={onClick}

   onKeyDown={(e) => e.key === 'Enter' && onClick()}
```

```
    >
      {label}
    </button>
  );
}
```

Enable React-Axe for Dev Testing:

```
npm install react-axe --save-dev
```

Then in your entry file:

```
if (process.env.NODE_ENV !== 'production') {
  const ReactAxe = require('react-axe');
  const ReactDOM = require('react-dom');
  ReactAxe(React, ReactDOM, 1000);
}
```

Use Cases

- Adding user settings to toggle high contrast or font scaling.
- Running React-Axe or Lighthouse to detect accessibility issues before deployment.
- Testing focus navigation across all UI components.