

Frontend LLM Testing Report

Table of Contents

1. [Introduction](#)
2. [Evaluation Methodology](#)
3. [Prompts](#)
 - o Given the description of a component and its props, create that component
 - o Given the implementation of a component, migrate its functionalities to using another dependency
 - o Given screenshots of a component, replicate the component
 - o Given the API endpoints, come up with a page that integrates these functionalities
 - o Given screenshots of a edit page of 24tracc, replicate its functionalities
4. [Test Cases](#)
5. [Results](#)
6. [Discussion](#)
7. [Summary](#)
8. [Appendix](#)
 - o Task 1
 - o Task 2
 - o Task 3
 - o Task 4
 - o Task 5

1. Introduction

The purpose of the evaluation is to compare the abilities of different modern LLM in frontend development situations. Each LLM model is prompted with 5 tasks with increasing difficulty. Results are gathered, analyzed (ran on local machine), and scored by our defined metrics.

2. Evaluation Methodology

Prompts are constructed in increasing difficulty based on 5 different situations that relates to frontend development:

- Creating a component given descriptions
- Migrating a component to a different external library
- Creating a component given a graphical design guide showing its states and appearance.
- Integrating functionalities given API endpoints
- Creating a page and relevant functionalities given a screenshot.

Each LLM in each situation will be strictly given only the prompt and the necessary file supplement, if there are any. The responses are scored based on the correctness and code quality related to the task.

3. Prompts:

Given the description of a component and its props, create that component.

Create a React functional component named `Button` with a CSS style file that meets the following requirements:

Props:

- `onClick`: function, called when the button is clicked (unless disabled).
- `disabled`: boolean, disables interaction and applies disabled styles.
- `size: "regular" or "large"`, sets height to 32px or 48px. Defaults to regular if not provided.
- `type: "primary", "secondary", or "warning"`, sets button color (`primary: #82a3ff 50%; secondary: #fcfcff 20%; warning: #b71818 50%`). Defaults to primary if not provided.

Children:

- String, rendered inside the button.

Styling:

- 20px horizontal inner padding, 8px border-radius.
- On hover, the button color should be less transparent.
- On click, the button color should be even less transparent and an inner shadow of `{ x: 4, y: 4, spread: 2, color: 000000, opacity: 20% }` should appear.
- When disabled: more transparent color.

Interactivity:

- No hover/click effects or `onClick` calls when disabled.

Given the implementation of a component, migrate its functionalities to using another dependency.

Create a React component with the same functionalities with the code below using Popper.js

```
import React from 'react';
import { Popover } from '@base-ui-components/react';

export type OPopoverProps = {
    align?: 'center' | 'start' | 'end';
    alignOffset?: number;
    side?: 'top' | 'bottom' | 'left' | 'right';
    sideOffset?: number;
    delay?: number;
    openOnHover?: boolean;
    open?: boolean;
    onOpenChange?: (
        open: boolean,
        event?: Event,
        reason?: Popover.Root.OpenChangeReason
    ) => void;
};

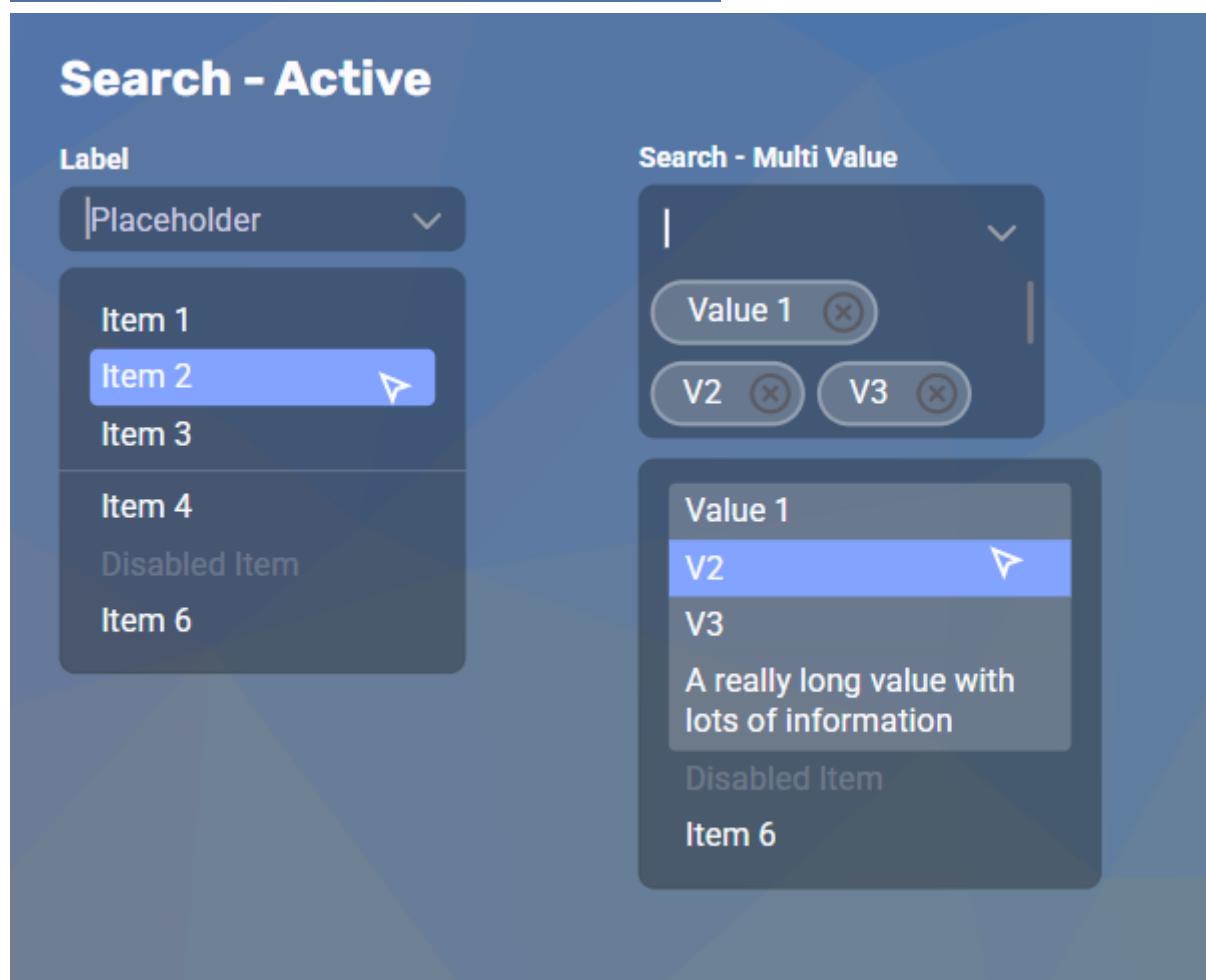
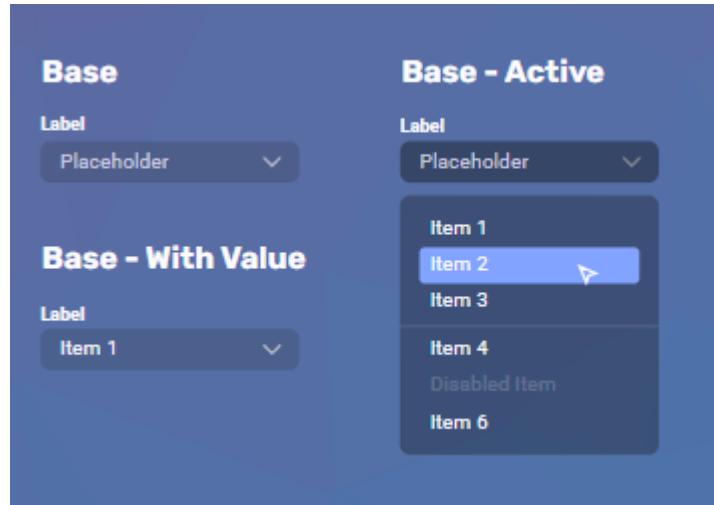
export function OPopover(props: React.PropsWithChildren<OPopoverProps>) {
    const getSlot = (slotName: string) =>
```

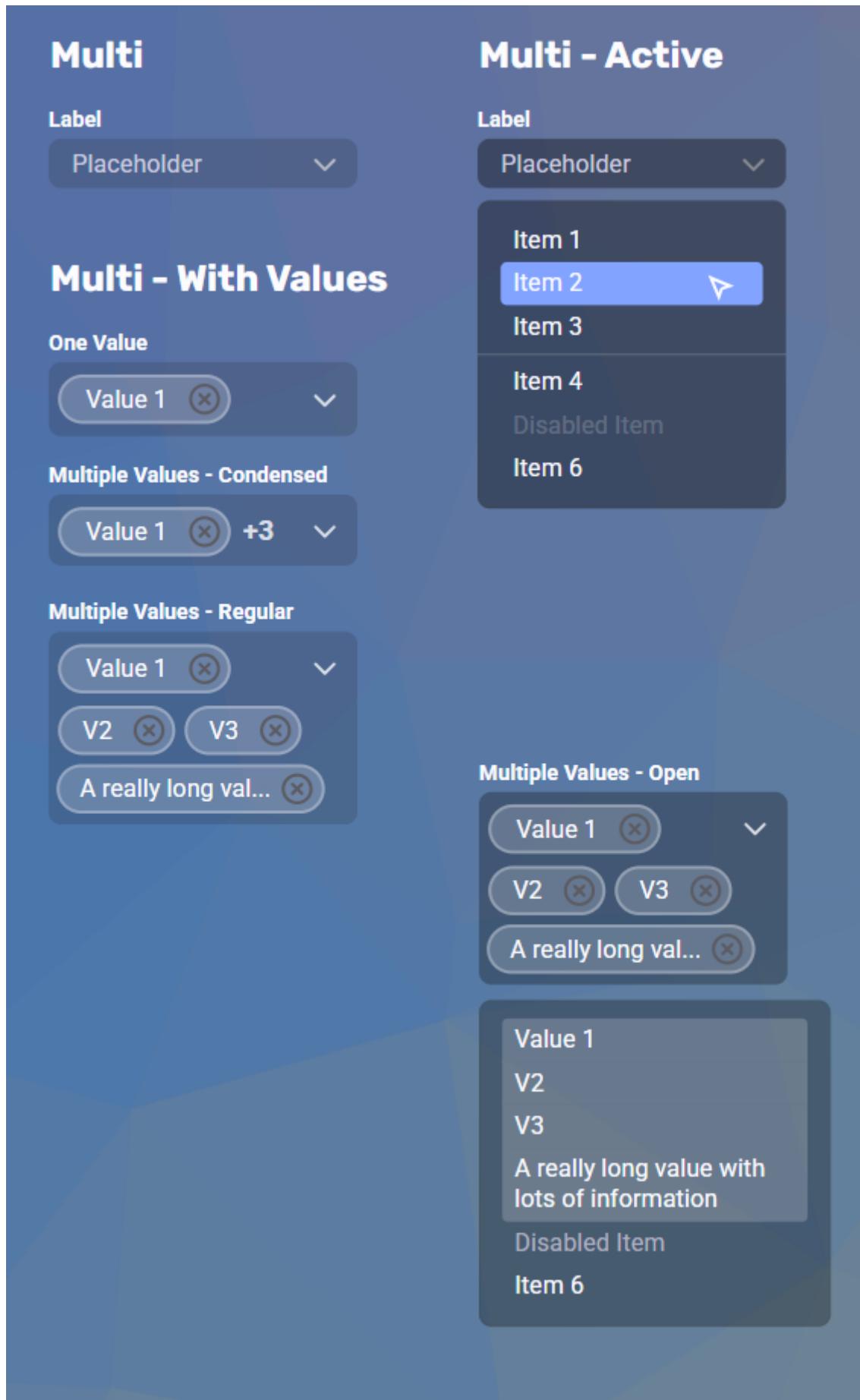
```
React.Children.toArray(props.children).find(  
  (child: React.ReactNode) =>  
    React.isValidElement(child) && (child.props as { slot?: string })?.slot === slotName  
);  
  
const title = getSlot('title') as string;  
const target = getSlot('target');  
  
const defaultContent = React.Children.toArray(props.children).filter(  
  (child: React.ReactNode) =>  
    !(React.isValidElement(child) && (child.props as { slot?: string })?.slot)  
);  
  
return (  
  <div className={styles.OPopover}>  
    <Popover.Root  
      open={props.open}  
      delay={props.delay ?? 350}  
      onOpenChange={props.onOpenChange}  
      openOnHover={props.openOnHover ?? false}>  
      <Popover.Trigger className={styles.OPopoverButton}>  
        {target}  
      </Popover.Trigger>  
      <Popover.Portal>  
        <Popover.Positioner  
          align={props.align ?? 'center'}  
          alignOffset={props.alignOffset ?? 0}  
          side={props.side ?? 'bottom'}  
          sideOffset={props.sideOffset ?? 0}>  
          <Popover.Popup  
            className={`  
              ${styles.OPopoverPopup}  
            `}>  
            <{title && (  
              <Popover.Title className={styles.OPopoverTitle}>  
                {title}  
              </Popover.Title>  
            )}>  
            <div className={styles.OPopoverContent}>  
              {defaultContent}  
            </div>  
          </Popover.Popup>  
        </Popover.Positioner>  
      </Popover.Portal>  
    </Popover.Root>  
  </div>  
);  
}
```

```
export default OPopover;
```

Given screenshots of a component, replicate the component

Given the following screenshots of a React dropdown component, replicate the React component with the states and the types shown





Given the API endpoints, come up with a page that integrates these functionalities

Given the API endpoints:

- **baseUrl**: http://localhost:8080
- **url**: /rest/product/search
- **method**: POST
- **body**:

```
{
    includeProductOriginData: false,
    includeSystemBulkUploads: false,
}
```

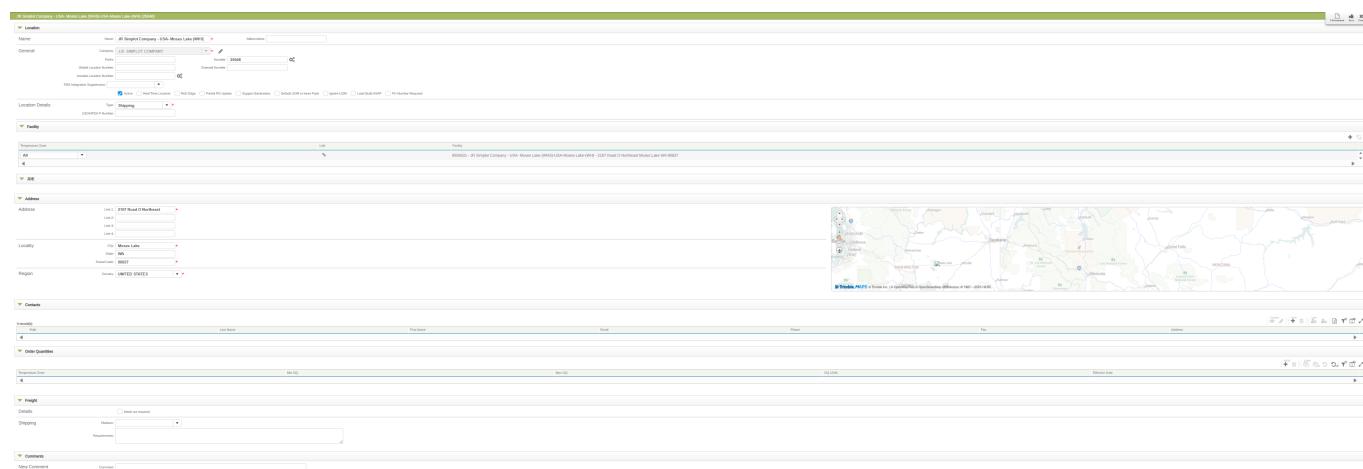
that returns { **data**: { [key: string]: any }[], **isLoading**: boolean }, and

- **baseUrl**: http://localhost:8080
- **url**: /rest/banners
- **method**: GET that returns { **id**: string, **message**: string, **style**: 'primary' | 'warning' | 'success' }

Develop a React webpage that displays the banners and the product data.

Given screenshots of a edit page of 24tracc, replicate its functionalities

Given the following screenshot of an edit page, replicate the React webpage with the functionalities:



4. Test cases

Given the description of a component and its props, create that component.

1. Component Structure

- The LLM should generate a React functional component named **Button**.
- The component should accept props: **text**, **onClick**, **disabled**, **size**, and **type**.

2. Props Functionality

- The **text** prop should be rendered inside the button.
- The **onClick** prop should be called when the button is clicked (if not disabled).
- The **disabled** prop should prevent interaction and apply disabled styles.

- The `size` prop should set height to 32px (regular) or 48px (large).
- The `type` prop should set the button color:
 - `primary`: #82a3ff, 50%
 - `secondary`: #fcfcff, 20%
 - `warning`: #b71818, 50%

3. Styling

- The button should have 20px horizontal inner padding and 8px border-radius.
- On hover, the button should be less opaque.
- On click, the button should be less opaque even further and an inner shadow of { x: 4, y: 4, spread: 2, color: 000000, opacity: 20% } should appear.
- When disabled, the button should be opaque and static in color.

4. Interactivity

- The button should not respond to hover or click events when disabled.
- The `onClick` function should not be called when the button is disabled.

5. Edge Cases

- If no `type` is provided, the button should use a default color.
- If no `size` is provided, the button should default to regular size.

6. Code Quality

- The generated code should be syntactically correct and follow React best practices.

Given the implementation of a component, migrate its functionalities to using another dependency.

1. Component Structure

- The LLM should generate a React functional component named `O-Popover`.
- The component should accept props: `align`, `alignOffset`, `side`, `sideOffset`, `delay`, `openOnHover`, `open`, and `onOpenChange`.

2. Props Functionality

- The `align` and `alignOffset` props should specify the alignment of the popover respective to the target.
- The `side` and `sideOffset` props should specify the side position of the popover respective to the target.
- The `delay` prop should specify the time taken, if any, for the popover to appear.
- The `openOnHover` prop should specify whether the popover opens on click or on hover.
- The `open` prop should specify whether the popover is open.
- The `onOpenChange` prop should be called when the popover is opened.
- The popover should fetch the children with slot `title` and apply a certain style to it.
- The popover should fetch the children with slot `target` and use that element as the target to generate the popover.
- The popover, if not `openOnHover`, should close when the user clicks outside of the popover.

3. Interactivity

- The popover should open on click-hover (specified by `openOnHover` prop) after a certain amount of time (specified by `delay`)

4. Edge Cases

- If no `align` is provided, the popover should align on center.
- If no `alignOffset` is provided, it should be 0.
- If no `side` is provided, the popover appear at the bottom.
- If no `sideOffset` is provided, it should be 0.
- If no `delay` is provided, the popover should appear after 350ms.
- If no `openOnHover` is provided, the popover should open on click, not on hover.

5. Code Quality

- The generated code should be syntactically correct and follow React best practices.

Given screenshots of a component, replicate the component

1. Component Structure

- The LLM should generate a React functional dropdown component.

2. Props Functionality

- The dropdown component should exhibit the following states:
 - `base`: displaying the placeholder
 - `with value`: displaying the value(s) selected.
 - `active`: the dropdown is focused.
- The dropdown component should exhibit the following types:
 - `base`: does not support search nor multi-value select.
 - A combination of:
 - `search`: the user is able to type in a partial search and selecting an option that includes the typed-in content.
 - `multiple`: the user is able to select multiple options from the menu; each selected option should be represented by a deletable tag that deselects the tag when deleted.
 - `condensed`: if the dropdown is of type `multiple`, it only displays the first selected option; the amount of the rest selected will be displayed.
- The user should be able to provide the options/dividers for the component.
- An option should not respond to user interaction if it is disabled.

3. Interactivity

- The dropdown component should exhibit the shown style change according to the functionality described above and shown in the images.

4. Edge Cases

- If dropdown is `multiple` and `condensed` and only one option is selected, it should not display the number of the other selected options.
- If dropdown is not `multiple` but `condensed`, the type `condensed` should have no effect on the functionality and the appearance of the component.
- If the content of the option is very long, its tag representation should hide the overflow and use ellipsis.

5. Code Quality

- The generated code should be syntactically correct and follow React best practices.

Given the API endpoints, come up with a page that integrates these functionalities

1. Component Structure

- The LLM should generate a working React webpage.

2. Props Functionality

- The webpage should fetch the products data given the API endpoint and display these data.
 - The webpage should take a reasonable amount of time to load the data.
 - A grid-like structure should be used to visualize the data.
- The webpage should fetch the banners data given the API endpoint and display the banners one at a time.
 - The user should be able to close the banners; alternatively, the banners can be automatically dismissed after a certain period of time.
 - The banner should exhibit distinguishable styles across the types.

3. Edge Cases

- The webpage should exhibit proper error handling if the APIs are unreachable.

4. Code Quality

- The generated code should be syntactically correct and follow React best practices.

Given screenshots of a edit page of 24tracc, replicate its functionalities

1. Component Structure

- The LLM should generate a working React webpage including the functionalities shown in the screenshot.

2. Props Functionality

- The user should be able to edit the information shown on the page. Once these changes are saved, they should see the changes on refresh.

3. Edge Cases

- The webpage should exhibit proper validation (type check / range check) against stupid input.

- When the changes are not saved, the original information should be displayed when the webpage reloads.

4. Code Quality

- The generated code should be syntactically correct and follow React best practices.

5. Results

The scores are given following this rule:

One strike is -1. A bigger strike is -2. 0 For not even trying.

Given the description of a component and its props, create that component.

GPT 4.1: 10/10

Same.dev: 10/10

Cursor: 5/10

- Text not aligned vertically in the center.
- Created drop shadow instead of inner shadow when the button is clicked.
- Created padding on all directions when only asked for horizontal padding, making the height wrong.
- Button color wrong when it is hovered: always using the same color even if the button type is secondary/warning.
- Use the same color for disabled primary, secondary, and warning buttons.

Claude Opus 4: 10 /10

- The only two models that requires `onClick` and changes the text color according to different `types`.

Claude Sonnet 4: 9/10

- The only two models that requires `onClick` and changes the text color according to different `types`.
- Integrated everything (Button, styles) into one file which can be messy to maintain.

Given the implementation of a component, migrate its functionalities to using another dependency.

GPT 4.1: 9/10

- Popovers will not close when clicked outside.

Same.dev: 10/10

Cursor: 6/10

- The name of the component is not OPopover.
- Popovers will not close when clicked outside.
- `delay` prop not working.
- Specifying `side` prop with `left` or `right` would result in the popover positioned to the far left/right of the viewport.

Claude Opus 4: 9/10

- Integrated everything (Popover, styles) into one file which can be messy to maintain.

Claude Sonnet 4: 8/10

- Integrated everything (Popover, styles) into one file which can be messy to maintain.
- Several TypeScript errors.

Given screenshots of a component, replicate the component

GPT 4.1: 2/10

- Dropdown root style missing `position: relative` so the dropdown menu is never shown.
- Didn't use React state for managing values, causing the selected value(s) never being displayed.
- No way to provide dividers.
- (Comparatively) Minor styling issues.

Same.dev: 7/10

- Style not exactly the same as shown in the images.
- No way to provide dividers.

Cursor: 3/10

- Style not exactly the same as shown in the images.
- No search dropdowns.
- No way to provide dividers.
- No condensed dropdowns.

Claude Opus 4: 6/10

- No way to provide dividers.
- Style not exactly the same as shown in the images.
- Cannot have searchable + multiple dropdown combination.

Claude Sonnet 4: 7/10

- Style not exactly the same as shown in the images.
- No way to provide dividers.

Given the API endpoints, come up with a page that integrates these functionalities

GPT 4.1: 7/10

- Banners are displayed merely as colored blocks
- Integrated everything (Product, Banner, styles) into one file which can be messy to maintain.

Same.dev: 8/10

- Banners are displayed merely as colored blocks.

Cursor: 6/10

- Banners are displayed merely as colored blocks.
- Product list unstyled.

Claude Opus 4: 7/10

- Banners are displayed merely as colored blocks
- Integrated everything (Product, Banner, styles) into one file which can be messy to maintain.

Claude Sonnet 4: 9/10

- Integrated everything (Components, styles) into one file which can be messy to maintain.

Given screenshots of a edit page of 24tracc, replicate its functionalities

GPT 4.1: 2/10

- Tracking Prevention blocked access to storage for GoogleMap URL. Map visualization not able to zoom.
- Add Contact button not working.
- Add Order Quantities button not working.
- No type checks on input values (validation).
- Several TypeScript Errors in code.
- Cannot download nor save changes.
- Integrated everything (Form, styles) into one file which can be messy to maintain.
- Layout not necessarily the best.

Same.dev: 2/10

- Appearance is very different than the one shown in the image.
- Buttons not working.
- No type checks (validation).
- Cannot download/save changes.

Cursor: 0/10

- Minimal effort given to replicate the page and style.

Claude Opus 4: 3/10

- No minimap loaded.
- Pretty much none of the buttons are working.
- Cannot save changes.
- Minimal type checks (validation).
- Appearance not exactly the same.

Claude Sonnet 4: 3/10

- No minimap loaded.
- Cannot save changes.
- No type checks (validation).
- Just missing a lot of stuff in general.

Scenario / Model	GPT 4.1	Same.dev	Cursor	Claude Opus 4	Claude Sonnet 4
Button	10	10	5	10	9
Popover Migration	9	10	6	9	8
Dropdown	2	7	3	6	7
Product Page API	7	8	6	7	9
Edit Page	2	2	0	3	3

6. Discussion

Model Performance

Claude Sonnet 4, Claude Opus 4, and Same.dev delivers high-quality code comparatively. These models tend to generate code that is functionally correct and better handles edge cases. GPT 4.1 performs well on the first task, which is designed to be straightforward, but is at most mediocre on more complex scenarios. Cursor is the model with the fastest response time; however, it often produces incomplete solutions and particularly struggles with advanced interactivity and styling.

Self-Correction and Previews

Besides that, Sonnet, Opus, and Same seems to pre-run the code after generating a version; in case they spot a mistake, they would make a newer version fixing it. They also provide a preview panel with a demonstration they generate for each scenario that the user can interact with. However, I assume this causes their tendency to put everything in a single file, making it harder to maintain in the future.

Common Weaknesses

Some common weaknesses can be seen across the five models. For example, banners are often rendered as simple colored blocks without the option to close them or for auto-dismissal. Dropdown and edit page tasks show that most models struggle with replicating UI with complex states and handling validation. This is particularly true when the models are fed with less specific prompts unlike task 1 has. They tend to perform poor when the prompt is as vague as "just generate me <something>."

Speed vs. Quality

The model that takes the longest time to generate a response is Same.dev, particularly because it would generate the components within a project setting (which, of course, is also generated and configured by the LLM). This feature could be useful at times.

On the other hand, Cursor really slays when we are talking about respond times. This comes at the cost of completeness and correctness, as sometimes it would just say "take this and add <these functionalities> by yourself and it would work" when the user expects it to generate complete code that meets the task requirements.

The Prompts and the Test Cases Themselves

As seen from the table above, the average score for task 4 is higher than task 3, which seems to show we should probably swap these tasks. However, these are my guesses on why this happens:

- Task 3 involves image recognition. Task 4 is just NLP.
- Task 3 focuses on building a component. Task 4 requires API integration. These are different focuses and it's hard to compare which one is more difficult than the other to the AI.

7. Summary

In summary, while all models except Cursor show promise, Sonnet, Opus, and Same really stand out on their ability to deal with frontend specific tasks. I would suggest prefer using these models for production UI work and use Cursor for quick prototyping.

Appendix:

Task 1

GPT 4.1:

Button.tsx

```
import React from "react";
import "./Button.css";

type ButtonProps = {
  onClick: () => void;
  disabled?: boolean;
  size?: "regular" | "large";
  type?: "primary" | "secondary" | "warning";
  children: string;
};

const Button: React.FC<ButtonProps> = ({
  onClick,
  disabled = false,
  size = "regular",
  type = "primary",
  children,
}) => {
  const className = [
    "btn",
    `btn--${type}`,
    `btn--${size}`,
    disabled ? "btn--disabled" : "",
  ]
    .filter(Boolean)
    .join(" ");

  return (
    <button
      className={className}
```

```
    onClick={disabled ? undefined : onClick}
    disabled={disabled}
    type="button"
  >
  {children}
</button>
);
};

export default Button;
```

Button.css

```
.btn {
  border: none;
  outline: none;
  padding: 0 20px;
  border-radius: 8px;
  font-size: 16px;
  cursor: pointer;
  transition:
    background 0.16s,
    box-shadow 0.16s;
  height: 32px;
  color: #222;
  position: relative;
  font-weight: 500;
  display: inline-flex;
  align-items: center;
  justify-content: center;
}

.btn--large {
  height: 48px;
  font-size: 18px;
}

.btn--primary {
  background: rgba(130, 163, 255, 0.5);
}

.btn--secondary {
  background: rgba(252, 252, 255, 0.2);
}

.btn--warning {
  background: rgba(183, 24, 24, 0.5);
}

/* Hover: less transparent */
.btn:not(.btn--disabled):hover {
  background: rgba(130, 163, 255, 0.7);
}

.btn--secondary:not(.btn--disabled):hover {
```

```
background: rgba(252, 252, 255, 0.35);
}

.btn--warning:not(.btn--disabled):hover {
  background: rgba(183, 24, 24, 0.7);
}

/* Active: even less transparent + shadow */
.btn:not(.btn--disabled):active {
  background: rgba(130, 163, 255, 0.85);
  box-shadow: 4px 4px 2px rgba(0, 0, 0, 0.20) inset;
}

.btn--secondary:not(.btn--disabled):active {
  background: rgba(252, 252, 255, 0.6);
  box-shadow: 4px 4px 2px rgba(0, 0, 0, 0.20) inset;
}

.btn--warning:not(.btn--disabled):active {
  background: rgba(183, 24, 24, 0.85);
  box-shadow: 4px 4px 2px rgba(0, 0, 0, 0.20) inset;
}

/* Disabled: opaque and static color */
.btn--disabled,
.btn:disabled {
  cursor: not-allowed;
  background: #ced3db;
  color: #888;
  opacity: 1;
  box-shadow: none;
}
```

Same.dev

Button.tsx

```
import type React from 'react';
import './Button.css';

interface ButtonProps {
  onClick?: () => void;
  disabled?: boolean;
  size?: 'regular' | 'large';
  type?: 'primary' | 'secondary' | 'warning';
  children: string;
}

const Button: React.FC<ButtonProps> = ({
  onClick,
  disabled = false,
  size = 'regular',
  type = 'primary',
  children
```

```
) => {
  const handleClick = () => {
    if (!disabled && onClick) {
      onClick();
    }
  };

  const className = `button button--${type} button--${size} ${disabled ? 'button--disabled' : ''}`.trim();

  return (
    <button
      className={className}
      onClick={handleClick}
      disabled={disabled}
    >
      {children}
    </button>
  );
};

export default Button;
```

Button.css

```
.button {
  border: none;
  border-radius: 8px;
  padding: 0 20px;
  cursor: pointer;
  font-family: inherit;
  font-size: 14px;
  font-weight: 500;
  transition: all 0.2s ease-in-out;
  outline: none;
  position: relative;
  box-sizing: border-box;
}

/* Size variants */
.button--regular {
  height: 32px;
}

.button--large {
  height: 48px;
  font-size: 16px;
}

/* Type variants */
.button--primary {
  background-color: rgba(130, 163, 255, 0.5);
```

```
    color: white;
}

.button--secondary {
  background-color: rgba(252, 252, 255, 0.2);
  color: #333;
}

.button--warning {
  background-color: rgba(183, 24, 24, 0.5);
  color: white;
}

/* Hover effects (only when not disabled) */
.button:not(.button--disabled):hover.button--primary {
  background-color: rgba(130, 163, 255, 0.7);
}

.button:not(.button--disabled):hover.button--secondary {
  background-color: rgba(252, 252, 255, 0.4);
}

.button:not(.button--disabled):hover.button--warning {
  background-color: rgba(183, 24, 24, 0.7);
}

/* Active/click effects (only when not disabled) */
.button:not(.button--disabled):active.button--primary {
  background-color: rgba(130, 163, 255, 0.9);
  box-shadow: inset 4px 4px 2px rgba(0, 0, 0, 0.2);
}

.button:not(.button--disabled):active.button--secondary {
  background-color: rgba(252, 252, 255, 0.6);
  box-shadow: inset 4px 4px 2px rgba(0, 0, 0, 0.2);
}

.button:not(.button--disabled):active.button--warning {
  background-color: rgba(183, 24, 24, 0.9);
  box-shadow: inset 4px 4px 2px rgba(0, 0, 0, 0.2);
}

/* Disabled state */
.button--disabled {
  cursor: not-allowed;
  opacity: 0.6;
}

.button--disabled.button--primary {
  background-color: rgba(130, 163, 255, 1);
}

.button--disabled.button--secondary {
  background-color: rgba(252, 252, 255, 1);
```

```
}

.button--disabled.button--warning {
  background-color: rgba(183, 24, 24, 1);
}
```

Cursor

Button.tsx

```
import React from 'react';
import './Button.css';

interface ButtonProps {
  onClick?: () => void;
  disabled?: boolean;
  size?: 'regular' | 'large';
  type?: 'primary' | 'secondary' | 'warning';
  children: string;
}

const Button: React.FC<ButtonProps> = ({  
  onClick,  
  disabled = false,  
  size = 'regular',  
  type = 'primary',  
  children,  
}) => {  
  const buttonClass = `button ${size} ${type} ${disabled ? 'disabled' : ''}`;  
  
  return (  
    <button  
      className={buttonClass}  
      onClick={!disabled ? onClick : undefined}  
      disabled={disabled}  
    >  
      {children}  
    </button>  
  );  
};  
  
export default Button;
```

Button.css

```
.button {  
  padding: 20px;  
  border-radius: 8px;  
  border: none;
```

```
cursor: pointer;
transition: background-color 0.3s, box-shadow 0.3s;
}

.button.regular {
  height: 32px;
}

.button.large {
  height: 48px;
}

.button.primary {
  background-color: rgba(130, 163, 255, 0.5);
}

.button.secondary {
  background-color: rgba(252, 252, 255, 0.2);
}

.button.warning {
  background-color: rgba(183, 24, 24, 0.5);
}

.button:hover:not(.disabled) {
  background-color: rgba(130, 163, 255, 0.7); /* Adjust for primary */
}

.button:active:not(.disabled) {
  background-color: rgba(130, 163, 255, 0.9); /* Adjust for primary */
  box-shadow: 4px 4px 2px rgba(0, 0, 0, 0.2);
}

.button.disabled {
  background-color: rgba(130, 163, 255, 0.3); /* Adjust for primary */
  cursor: not-allowed;
}
```

Claude Opus 4

Button.tsx

```
import React from 'react';
import './Button.css';

const Button = ({
  onClick,
  disabled = false,
  size = 'regular',
  type = 'primary',
  children
```

```
) => {
  const handleClick = (e) => {
    if (!disabled && onClick) {
      onClick(e);
    }
  };

  const className = `button button--${type} button--${size} ${disabled ? 'button--disabled' : ''}`;

  return (
    <button
      className={className}
      onClick={handleClick}
      disabled={disabled}
    >
      {children}
    </button>
  );
};

export default Button;
```

Buttons.css

```
.button {
  padding: 0 20px;
  border-radius: 8px;
  border: none;
  cursor: pointer;
  font-family: inherit;
  font-size: 14px;
  transition: all 0.2s ease;
  outline: none;
  position: relative;
}

/* Size variants */
.button--regular {
  height: 32px;
}

.button--large {
  height: 48px;
  font-size: 16px;
}

/* Type variants */
.button--primary {
  background-color: rgba(130, 163, 255, 0.5);
  color: #1a1a1a;
}
```

```
.button--secondary {
  background-color: rgba(252, 252, 255, 0.2);
  color: #1a1a1a;
  border: 1px solid rgba(0, 0, 0, 0.1);
}

.button--warning {
  background-color: rgba(183, 24, 24, 0.5);
  color: white;
}

/* Hover states */
.button:not(.button--disabled):hover {
  cursor: pointer;
}

.button--primary:not(.button--disabled):hover {
  background-color: rgba(130, 163, 255, 0.7);
}

.button--secondary:not(.button--disabled):hover {
  background-color: rgba(252, 252, 255, 0.4);
}

.button--warning:not(.button--disabled):hover {
  background-color: rgba(183, 24, 24, 0.7);
}

/* Active states */
.button:not(.button--disabled):active {
  box-shadow: inset 4px 4px 2px rgba(0, 0, 0, 0.2);
}

.button--primary:not(.button--disabled):active {
  background-color: rgba(130, 163, 255, 0.9);
}

.button--secondary:not(.button--disabled):active {
  background-color: rgba(252, 252, 255, 0.6);
}

.button--warning:not(.button--disabled):active {
  background-color: rgba(183, 24, 24, 0.9);
}

/* Disabled states */
.button--disabled {
  cursor: not-allowed;
  opacity: 0.5;
}

.button--primary.button--disabled {
  background-color: rgba(130, 163, 255, 0.3);
```

```
}

.button--secondary.button--disabled {
  background-color: rgba(252, 252, 255, 0.1);
}

.button--warning.button--disabled {
  background-color: rgba(183, 24, 24, 0.3);
}
```

Claude Sonnet 4

Button.tsx

```
import { useState } from 'react';

const Button = ({
  onClick,
  disabled = false,
  size = "regular",
  type = "primary",
  children
}) => {
  const [isPressed, setIsPressed] = useState(false);

  const handleClick = () => {
    if (!disabled && onClick) {
      onClick();
    }
  };

  const handleMouseDown = () => {
    if (!disabled) {
      setIsPressed(true);
    }
  };

  const handleMouseUp = () => {
    setIsPressed(false);
  };

  const handleMouseLeave = () => {
    setIsPressed(false);
  };

  // Base styles
  const baseStyles = {
    padding: '0 20px',
    borderRadius: '8px',
    border: 'none',
  }
```

```
cursor: disabled ? 'not-allowed' : 'pointer',
fontSize: '14px',
fontWeight: '500',
transition: 'all 0.15s ease',
outline: 'none',
userSelect: 'none'
};

// Size styles
const sizeStyles = {
  regular: { height: '32px' },
  large: { height: '48px' }
};

// Color styles based on type and state
const getBackgroundColor = () => {
  const colors = {
    primary: '#82a3ff',
    secondary: '#fcfcff',
    warning: '#b71818'
  };

  const baseColor = colors[type];

  if (disabled) {
    return `${baseColor}20`; // 20% opacity for disabled
  }

  if (isPressed) {
    return `${baseColor}80`; // 80% opacity for pressed
  }

  return `${baseColor}80`; // 50% opacity for normal (using 80 hex value for
50%)
};

// Box shadow for pressed state
const boxShadow = isPressed && !disabled
  ? 'inset 4px 4px 2px rgba(0, 0, 0, 0.2)'
  : 'none';

const combinedStyles = {
  ...baseStyles,
  ...sizeStyles[size],
  backgroundColor: getBackgroundColor(),
  boxShadow,
  color: type === 'secondary' ? '#333' : 'fff'
};

// Hover styles
const hoverStyles = !disabled ? {
  ':hover': {
    backgroundColor: type === 'primary' ? '#82a3ffb3' :
      type === 'secondary' ? '#fcfcff40' :

```

```
        '#b71818B3'
    }
} : {};

return (
  <button
    style={combinedStyles}
    onClick={handleClick}
    onMouseDown={handleMouseDown}
    onMouseUp={handleMouseUp}
    onMouseLeave={handleMouseLeave}
    disabled={disabled}
    onMouseEnter={(e) => {
      if (!disabled) {
        const colors = {
          primary: '#82a3ffB3',
          secondary: '#fcfcff40',
          warning: '#b71818B3'
        };
        e.target.style.backgroundColor = colors[type];
      }
    }}
    onMouseLeave={(e) => {
      if (!disabled && !isPressed) {
        e.target.style.backgroundColor = getBackgroundColor();
      }
    }}
  >
  {children}
  </button>
);
};
```

Task 2

GPT 4.1

OPopover.tsx

```
import React, { useRef, useState, useEffect } from "react";
import { createPopper, Placement } from "@popperjs/core";
import styles from "./OPopover.module.css";

export type OPopoverProps = {
  align?: "center" | "start" | "end";
  alignOffset?: number;
  side?: "top" | "bottom" | "left" | "right";
  sideOffset?: number;
  delay?: number;
  openOnHover?: boolean;
  open?: boolean;
```

```
onOpenChange?: (open: boolean, event?: Event | React.MouseEvent | React.KeyboardEvent, reason?: "manual" | "hover" | "focus" | "escape") => void;
};

function getSlot(children: React.ReactNode, slotName: string) {
  return React.Children.toArray(children).find(
    (child: React.ReactNode) =>
      React.isValidElement(child) && (child.props as { slot?: string }).slot === slotName
  );
}

function getDefaultContent(children: React.ReactNode) {
  return React.Children.toArray(children).filter(
    (child: React.ReactNode) =>
      !(React.isValidElement(child) && (child.props as { slot?: string }).slot)
  );
}

export function OPopover(props: React.PropsWithChildren<OPopoverProps>) {
  const {
    align = "center",
    alignOffset = 0,
    side = "bottom",
    sideOffset = 0,
    delay = 350,
    openOnHover = false,
    open: controlledOpen,
    onOpenChange,
    children,
  } = props;

  const [isOpen, setIsOpen] = useState(false);
  const triggerRef = useRef<HTMLButtonElement | null>(null);
  const popperRef = useRef<HTMLDivElement | null>(null);
  const popperInstance = useRef<ReturnType<typeof createPopper> | null>(null);
  const hoverTimeout = useRef<NodeJS.Timeout | null>(null);

  // Controlled/uncontrolled open state
  const isPopoverOpen = controlledOpen ?? isOpen;

  // Slot handling
  const target = getSlot(children, "target");
  const title = getSlot(children, "title");
  const defaultContent = getDefaultContent(children);

  // Placement mapping
  const placement: Placement = () => {
    let main = side;
    let alignMap: Record<string, string> = {
      start: "start",
      end: "end",
      bottom: "top",
      top: "bottom",
      left: "right",
      right: "left",
    };
    if (alignOffset) {
      main += alignOffset > 0 ? "-end" : "start";
    }
    if (sideOffset) {
      main += sideOffset > 0 ? "-bottom" : "top";
    }
    if (align) {
      main += align;
    }
    return main;
  };
}
```

```
        end: "end",
        center: "",
    );
const alignStr = alignMap[align] ? `-${alignMap[align]}` : "";
return (main + alignStr) as Placement;
})();

// Popper setup
useEffect(() => {
    if (isPopoverOpen && triggerRef.current && popperRef.current) {
        popperInstance.current = createPopper(triggerRef.current, popperRef.current,
{
    placement,
    modifiers: [
        {
            name: "offset",
            options: {
                offset: [alignOffset, sideOffset],
            },
        },
    ],
});
    }
    return () => {
        if (popperInstance.current) {
            popperInstance.current.destroy();
            popperInstance.current = null;
        }
    };
});
// eslint-disable-next-line
}, [isPopoverOpen, placement, alignOffset, sideOffset]);

// Open/close handlers
const handleOpenChange = (nextOpen: boolean, event?: any, reason?: any) => {
    if (controlledOpen === undefined) setIsOpen(nextOpen);
    onOpenChange?.(nextOpen, event, reason);
};

// Hover logic
const handleTriggerMouseEnter = (e: React.MouseEvent) => {
    if (!openOnHover || controlledOpen !== undefined) return;
    hoverTimeout.current = setTimeout(() => {
        handleOpenChange(true, e, "hover");
    }, delay);
};

const handleTriggerMouseLeave = (e: React.MouseEvent) => {
    if (!openOnHover || controlledOpen !== undefined) return;
    if (hoverTimeout.current) clearTimeout(hoverTimeout.current);
    handleOpenChange(false, e, "hover");
};

// Click logic
const handleTriggerClick = (e: React.MouseEvent) => {
```

```
if (openOnHover || controlledOpen !== undefined) return;
handleOpenChange(!isPopoverOpen, e, "manual");
};

// Escape key logic
useEffect(() => {
  if (!isPopoverOpen) return;
  const handleKeyDown = (e: KeyboardEvent) => {
    if (e.key === "Escape") {
      handleOpenChange(false, e, "escape");
    }
  };
  document.addEventListener("keydown", handleKeyDown);
  return () => document.removeEventListener("keydown", handleKeyDown);
  // eslint-disable-next-line
}, [isPopoverOpen]);

return (
  <div className={styles.OPopover}>
    <button
      ref={triggerRef}
      className={styles.OPopoverButton}
      onMouseEnter={handleTriggerMouseEnter}
      onMouseLeave={handleTriggerMouseLeave}
      onClick={handleTriggerClick}
      aria-haspopup="dialog"
      aria-expanded={isPopoverOpen}
      type="button"
    >
      {target}
    </button>
    {isPopoverOpen && (
      <div
        ref={popperRef}
        className={styles.OPopoverPopup}
        role="dialog"
        onMouseEnter={() => {
          if (openOnHover && hoverTimeout.current)
            clearTimeout(hoverTimeout.current);
        }}
        onMouseLeave={handleTriggerMouseLeave}
      >
        {title && (
          <div className={styles.OPopoverTitle}>
            {title}
          </div>
        )}
        <div className={styles.OPopoverContent}>
          {defaultContent}
        </div>
      </div>
    )}
  </div>
);
```

```
}

export default OPopover;
```

OPopover.module.css

```
.OPopover {
  display: inline-block;
  position: relative;
}

.OPopoverButton {
  background: none;
  border: none;
  cursor: pointer;
  padding: 0;
  font: inherit;
}

.OPopoverPopup {
  background: #fff;
  border: 1px solid #ececfc;
  border-radius: 8px;
  box-shadow: 0 4px 24px rgba(0,0,0,0.08);
  min-width: 200px;
  z-index: 1000;
  padding: 12px;
}

.OPopoverTitle {
  font-weight: bold;
  margin-bottom: 8px;
  font-size: 16px;
}

.OPopoverContent {
  font-size: 14px;
}
```

Same.dev

OPopover.tsx

```
import React, { useState, useRef, useEffect, useCallback } from 'react';
import { usePopper } from 'react-popper';
import { createPortal } from 'react-dom';
import type { Placement } from '@popperjs/core';
import './OPopover.css';
```

```
export type OPopoverProps = {
  align?: 'center' | 'start' | 'end';
  alignOffset?: number;
  side?: 'top' | 'bottom' | 'left' | 'right';
  sideOffset?: number;
  delay?: number;
  openOnHover?: boolean;
  open?: boolean;
  onOpenChange?: (
    open: boolean,
    event?: Event,
    reason?: 'click' | 'hover' | 'focus' | 'escape'
  ) => void;
};

export function OPopover(props: React.PropsWithChildren<OPopoverProps>) {
  const [isOpen, setIsOpen] = useState(props.open ?? false);
  const [referenceElement, setReferenceElement] = useState<HTMLElement | null>(null);
  const [popperElement, setPopperElement] = useState<HTMLElement | null>(null);
  const timeoutRef = useRef<number | null>(null);
  const isControlled = props.open !== undefined;

  // Convert props to Popper placement
  const getPlacement = (): Placement => {
    const side = props.side ?? 'bottom';
    const align = props.align ?? 'center';

    if (align === 'center') return side as Placement;
    return `${side}-${align}` as Placement;
  };

  const { styles, attributes } = usePopper(referenceElement, popperElement, {
    placement: getPlacement(),
    modifiers: [
      {
        name: 'offset',
        options: {
          offset: [props.alignOffset ?? 0, props.sideOffset ?? 0],
        },
      },
      {
        name: 'preventOverflow',
        options: {
          padding: 8,
        },
      },
    ],
  });
}

// Handle controlled vs uncontrolled state
const open = isControlled ? (props.open ?? false) : isOpen;

const handleOpenChange = useCallback((newOpen: boolean, event?: Event, reason?:
```

```
'click' | 'hover' | 'focus' | 'escape') => {
  if (!isControlled) {
    setIsOpen(newOpen);
  }
  props.onOpenChange?.(newOpen, event, reason);
}, [isControlled, props.onOpenChange]);
```

```
// Clear timeout on unmount
useEffect(() => {
  return () => {
    if (timeoutRef.current) {
      clearTimeout(timeoutRef.current);
    }
  };
}, []);
```

```
// Update internal state when controlled prop changes
useEffect(() => {
  if (isControlled) {
    setIsOpen(props.open ?? false);
  }
}, [props.open, isControlled]);
```

```
// Hover handlers
const handleMouseEnter = (event: React.MouseEvent) => {
  if (!props.openOnHover) return;

  if (timeoutRef.current) {
    clearTimeout(timeoutRef.current);
  }

  const delay = props.delay ?? 350;
  timeoutRef.current = window.setTimeout(() => {
    handleOpenChange(true, event.nativeEvent, 'hover');
  }, delay);
};
```

```
const handleMouseLeave = (event: React.MouseEvent) => {
  if (!props.openOnHover) return;

  if (timeoutRef.current) {
    clearTimeout(timeoutRef.current);
  }

  handleOpenChange(false, event.nativeEvent, 'hover');
};
```

```
// Click handler
const handleClick = (event: React.MouseEvent) => {
  if (props.openOnHover) return; // Don't handle clicks when hover is enabled

  handleOpenChange(!open, event.nativeEvent, 'click');
};
```

```
// Escape key handler
useEffect(() => {
  const handleEscape = (event: KeyboardEvent) => {
    if (event.key === 'Escape' && open) {
      handleOpenChange(false, event, 'escape');
    }
  };

  if (open) {
    document.addEventListener('keydown', handleEscape);
    return () => document.removeEventListener('keydown', handleEscape);
  }
}, [open, handleOpenChange]);

// Click outside handler
useEffect(() => {
  const handleClickOutside = (event: MouseEvent) => {
    if (
      open &&
      referenceElement &&
      popperElement &&
      !referenceElement.contains(event.target as Node) &&
      !popperElement.contains(event.target as Node)
    ) {
      handleOpenChange(false, event, 'click');
    }
  };

  if (open) {
    document.addEventListener('mousedown', handleClickOutside);
    return () => document.removeEventListener('mousedown', handleClickOutside);
  }
}, [open, referenceElement, popperElement, handleOpenChange]);

// Slot-based children extraction
const getSlot = (slotName: string) =>
  React.Children.toArray(props.children).find(
    (child: React.ReactNode) =>
      React.isValidElement(child) && (child.props as { slot?: string })?.slot
      === slotName
  );

const title = getSlot('title');
const target = getSlot('target');

const defaultContent = React.Children.toArray(props.children).filter(
  (child: React.ReactNode) =>
    !(React.isValidElement(child) && (child.props as { slot?: string })?.slot)
);

const popupContent = (
  <div
    ref={setPopperElement}
    style={styles.popper}
```

```
{...attributes.popper}
className={`OPopoverPopup ${open ? 'OPopoverPopup--open' : ''}`}
onMouseEnter={props.openOnHover ? handleMouseEnter : undefined}
onMouseLeave={props.openOnHover ? handleMouseLeave : undefined}
>
{title && (
  <div className="OPopoverTitle">
    {title}
  </div>
)
}

<div className="OPopoverContent">
  {defaultContent}
</div>
</div>
);

return (
<div className="OPopover">
<div
  ref={setReferenceElement}
  className="OP popoverButton"
  onClick={handleClick}
  onMouseEnter={props.openOnHover ? handleMouseEnter : undefined}
  onMouseLeave={props.openOnHover ? handleMouseLeave : undefined}
>
  {target}
</div>

{open && createPortal(popupContent, document.body)}
</div>
);
}

export default OPopover;
```

OPopover.css

```
.OPopover {
  display: inline-block;
}

.OP popoverButton {
  cursor: pointer;
  display: inline-block;
}

.OP popoverPopup {
  background: white;
  border: 1px solid #e1e5e9;
  border-radius: 8px;
  box-shadow: 0 4px 16px rgba(0, 0, 0, 0.12);
```

```
z-index: 1000;
max-width: 320px;
min-width: 200px;
opacity: 0;
transform: scale(0.95);
transition: opacity 0.2s ease-in-out, transform 0.2s ease-in-out;
pointer-events: none;
}

.OPopoverPopup--open {
  opacity: 1;
  transform: scale(1);
  pointer-events: auto;
}

.OPopoverTitle {
  padding: 12px 16px 8px 16px;
  font-weight: 600;
  font-size: 14px;
  color: #1a1a1a;
  border-bottom: 1px solid #f0f0f0;
  margin-bottom: 8px;
}

.OPopoverContent {
  padding: 12px 16px;
  font-size: 14px;
  line-height: 1.5;
  color: #4a4a4a;
}

/* Arrow styles for different placements */
.OPopoverPopup[data-popper-placement^='top'] {
  margin-bottom: 8px;
}

.OPopoverPopup[data-popper-placement^='bottom'] {
  margin-top: 8px;
}

.OPopoverPopup[data-popper-placement^='left'] {
  margin-right: 8px;
}

.OPopoverPopup[data-popper-placement^='right'] {
  margin-left: 8px;
}

/* Focus styles for accessibility */
.OPopoverButton:focus-visible {
  outline: 2px solid #82a3ff;
  outline-offset: 2px;
}
```

Cursor

CustomPopover.tsx

```
import React, { useState, useEffect, useRef } from 'react';
import { createPopper } from '@popperjs/core';
import './CustomPopover.css';

export type CustomPopoverProps = {
  align?: 'center' | 'start' | 'end';
  alignOffset?: number;
  side?: 'top' | 'bottom' | 'left' | 'right';
  sideOffset?: number;
  delay?: number;
  openOnHover?: boolean;
  open?: boolean;
  onOpenChange?: (open: boolean, event?: Event) => void;
};

export function CustomPopover(props: React.PropsWithChildren<CustomPopoverProps>)
{
  const [isOpen, setIsOpen] = useState(props.open || false);
  const popoverRef = useRef<HTMLDivElement>(null);
  const targetRef = useRef<HTMLButtonElement>(null);

  const handleOpenChange = (open: boolean, event?: Event) => {
    setIsOpen(open);
    if (props.onOpenChange) {
      props.onOpenChange(open, event);
    }
  };

  useEffect(() => {
    if (isOpen && targetRef.current && popoverRef.current) {
      createPopper(targetRef.current, popoverRef.current, {
        placement: props.side || 'bottom',
        modifiers: [
          {
            name: 'offset',
            options: {
              offset: [0, props.sideOffset || 0],
            },
          },
        ],
      });
    }
  }, [isOpen, props.side, props.sideOffset]);

  const handleMouseEnter = () => {
    if (props.openOnHover) {
      handleOpenChange(true);
    }
  };
}
```

```
        }

    };

    const handleMouseLeave = () => {
        if (props.openOnHover) {
            handleOpenChange(false);
        }
    };

    const title = React.Children.toArray(props.children).find(
        (child: React.ReactNode) =>
            React.isValidElement(child) && (child.props as { slot?: string }).slot ===
        'title'
    ) as string;

    const target = React.Children.toArray(props.children).find(
        (child: React.ReactNode) =>
            React.isValidElement(child) && (child.props as { slot?: string }).slot ===
        'target'
    );

    const defaultContent = React.Children.toArray(props.children).filter(
        (child: React.ReactNode) =>
            !(React.isValidElement(child) && (child.props as { slot?: string }).slot)
    );
}

return (
    <div className="custom-popover">
        <div
            ref={targetRef}
            onMouseEnter={handleMouseEnter}
            onMouseLeave={handleMouseLeave}
            className="custom-popover-button"
            onClick={() => handleOpenChange(!isOpen)}
        >
            {target}
        </div>
        {isOpen && (
            <div ref={popoverRef} className="custom-popover-popup">
                {title && <div className="custom-popover-title">{title}</div>}
                <div className="custom-popover-content">{defaultContent}</div>
            </div>
        )}
    </div>
);
};

export default CustomPopover;
```

CustomPopover.css

```
.custom-popover {  
  position: relative;  
}  
  
.custom-popover-button {  
  cursor: pointer;  
}  
  
.custom-popover-popup {  
  background-color: white;  
  border: 1px solid #ccc;  
  border-radius: 4px;  
  padding: 10px;  
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);  
  z-index: 1000;  
}  
  
.custom-popover-title {  
  font-weight: bold;  
  margin-bottom: 5px;  
}  
  
.custom-popover-content {  
  /* Additional styles for content */  
}
```

Claude Opus 4

OPopover.tsx

```
import React, { useState, useRef, useEffect, useCallback } from 'react';  
import { usePopper } from 'react-popper';  
  
export type OPopoverProps = {  
  align?: 'center' | 'start' | 'end';  
  alignOffset?: number;  
  side?: 'top' | 'bottom' | 'left' | 'right';  
  sideOffset?: number;  
  delay?: number;  
  openOnHover?: boolean;  
  open?: boolean;  
  onOpenChange?: (  
    open: boolean,  
    event?: Event,  
    reason?: 'click' | 'hover' | 'focus' | 'blur' | 'escape'  
  ) => void;  
};  
  
export function OPopover(props: React.PropsWithChildren<OPopoverProps>) {
```

```
const [isOpen, setIsOpen] = useState(props.open ?? false);
const [referenceElement, setReferenceElement] = useState<HTMLElement | null>(null);
const [popperElement, setPopperElement] = useState<HTMLElement | null>(null);
const timeoutRef = useRef<NodeJS.Timeout>();

// Get slots from children
const getSlot = (slotName: string) =>
  React.Children.toArray(props.children).find(
    (child: React.ReactNode) =>
      React.isValidElement(child) && (child.props as { slot?: string })?.slot
    === slotName
  );

const title = getSlot('title') as React.ReactElement;
const target = getSlot('target');

const defaultContent = React.Children.toArray(props.children).filter(
  (child: React.ReactNode) =>
    !(React.isValidElement(child) && (child.props as { slot?: string })?.slot)
);

// Map alignment values
const getPlacement = () => {
  const sideMap = {
    top: 'top',
    bottom: 'bottom',
    left: 'left',
    right: 'right'
  };

  const alignMap = {
    start: '-start',
    end: '-end',
    center: ''
  };
}

return `${sideMap[props.side ?? 'bottom']} ${alignMap[props.align ?? 'center']}` as any;
};

// Configure Popper
const { styles, attributes } = usePopper(referenceElement, popperElement, {
  placement: getPlacement(),
  modifiers: [
    {
      name: 'offset',
      options: {
        offset: [props.alignOffset ?? 0, props.sideOffset ?? 0],
      },
    },
    {
      name: 'preventOverflow',
      options: {
        boundary: 'viewport',
        padding: 8,
      },
    },
  ],
});
```

```
        padding: 8,
    },
},
],
});
};

// Handle controlled/uncontrolled state
useEffect(() => {
    if (props.open !== undefined) {
        setIsOpen(props.open);
    }
}, [props.open]);

const handleOpenChange = useCallback((open: boolean, event?: Event, reason?: 'click' | 'hover' | 'focus' | 'blur' | 'escape') => {
    if (props.open === undefined) {
        setIsOpen(open);
    }
    props.onOpenChange?.(open, event, reason);
}, [props.open, props.onOpenChange]);

// Click handlers
const handleTriggerClick = (e: React.MouseEvent) => {
    if (!props.openOnHover) {
        handleOpenChange(!isOpen, e.nativeEvent, 'click');
    }
};

// Hover handlers
const handleMouseEnter = (e: React.MouseEvent) => {
    if (props.openOnHover) {
        clearTimeout(timeoutRef.current);
        timeoutRef.current = setTimeout(() => {
            handleOpenChange(true, e.nativeEvent, 'hover');
        }, props.delay ?? 350);
    }
};

const handleMouseLeave = (e: React.MouseEvent) => {
    if (props.openOnHover) {
        clearTimeout(timeoutRef.current);
        timeoutRef.current = setTimeout(() => {
            handleOpenChange(false, e.nativeEvent, 'hover');
        }, 100);
    }
};

// Escape key handler
useEffect(() => {
    const handleEscape = (e: KeyboardEvent) => {
        if (e.key === 'Escape' && isOpen) {
            handleOpenChange(false, e, 'escape');
        }
    };
});
```

```
if (isOpen) {
  document.addEventListener('keydown', handleEscape);
  return () => document.removeEventListener('keydown', handleEscape);
}
}, [isOpen, handleOpenChange]);

// Click outside handler
useEffect(() => {
  const handleClickOutside = (e: MouseEvent) => {
    if (
      isOpen &&
      !props.openOnHover &&
      referenceElement &&
      popperElement &&
      !referenceElement.contains(e.target as Node) &&
      !popperElement.contains(e.target as Node)
    ) {
      handleOpenChange(false, e, 'click');
    }
  };
}

if (isOpen) {
  document.addEventListener('mousedown', handleClickOutside);
  return () => document.removeEventListener('mousedown', handleClickOutside);
}
}, [isOpen, props.openOnHover, referenceElement, popperElement,
handleOpenChange]);

// Cleanup timeout on unmount
useEffect(() => {
  return () => {
    clearTimeout(timeoutRef.current);
  };
}, []);

return (
  <div className="o-popover">
    <div
      ref={setReferenceElement}
      className="o-popover-trigger"
      onClick={handleTriggerClick}
      onMouseEnter={handleMouseEnter}
      onMouseLeave={handleMouseLeave}
      style={{ display: 'inline-block', cursor: 'pointer' }}
    >
      {target}
    </div>

    {isOpen && (
      <div
        ref={setPopperElement}
        style={styles.popper}
        {...attributes.popper}
    
```

```

        className="o-popover-popup"
        onMouseEnter={props.openOnHover ? handleMouseEnter : undefined}
        onMouseLeave={props.openOnHover ? handleMouseLeave : undefined}
      >
      <div
        style={{
          backgroundColor: 'white',
          border: '1px solid #e0e0e0',
          borderRadius: '8px',
          boxShadow: '0 2px 8px rgba(0, 0, 0, 0.15)',
          padding: '12px',
          minWidth: '150px',
          maxWidth: '350px',
          zIndex: 1000,
        }}
      >
      {title && (
        <div
          className="o-popover-title"
          style={{
            fontWeight: 600,
            marginBottom: '8px',
            fontSize: '14px',
            color: '#333',
          }}
        >
          {title}
        </div>
      )}
      <div className="o-popover-content" style={{ fontSize: '14px', color: '#666' }}>
        {defaultContent}
      </div>
    </div>
  </div>
)
</div>
);
}

```

Claude Sonnet 4

OPopover.tsx

```

import React, { useState, useRef, useEffect } from 'react';
import { createPopper } from '@popperjs/core';

export type OPopoverProps = {
  align?: 'center' | 'start' | 'end';
  alignOffset?: number;

```

```
side?: 'top' | 'bottom' | 'left' | 'right';
sideOffset?: number;
delay?: number;
openOnHover?: boolean;
open?: boolean;
onOpenChange?: (
  open: boolean,
  event?: Event,
  reason?: 'click' | 'hover' | 'focus' | 'escape' | 'outside-click'
) => void;
};

export function OPopover(props: React.PropsWithChildren<OPopoverProps>) {
  const [isOpen, setIsOpen] = useState(props.open ?? false);
  const [popperInstance, setPopperInstance] = useState(null);
  const triggerRef = useRef(null);
  const popupRef = useRef(null);
  const hoverTimeoutRef = useRef(null);
  const delayTimeoutRef = useRef(null);

  // Extract slotted children
  const getSlot = (slotName: string) =>
    React.Children.toArray(props.children).find(
      (child: React.ReactNode) =>
        React.isValidElement(child) && (child.props as { slot?: string })?.slot
      === slotName
    );

  const title = getSlot('title');
  const target = getSlot('target');

  const defaultContent = React.Children.toArray(props.children).filter(
    (child: React.ReactNode) =>
      !(React.isValidElement(child) && (child.props as { slot?: string })?.slot)
  );

  // Convert align prop to Popper.js offset
  const getPopperPlacement = () => {
    const side = props.side ?? 'bottom';
    const align = props.align ?? 'center';

    if (align === 'center') return side;
    return `${side}-${align}`;
  };

  // Handle controlled/uncontrolled state
  const actualIsOpen = props.open !== undefined ? props.open : isOpen;

  const handleOpenChange = (open: boolean, event?: Event, reason?: string) => {
    if (props.open === undefined) {
      setIsOpen(open);
    }
    props.onOpenChange?(open, event, reason as any);
  };
}
```

```
// Create/destroy popper instance
useEffect(() => {
  if (actualIsOpen && triggerRef.current && popupRef.current) {
    const modifiers = [
      {
        name: 'offset',
        options: {
          offset: [props.alignOffset ?? 0, props.sideOffset ?? 0],
        },
      },
      {
        name: 'preventOverflow',
        options: {
          padding: 8,
        },
      },
      {
        name: 'flip',
        options: {
          fallbackPlacements: ['top', 'bottom', 'left', 'right'],
        },
      },
    ];
    const instance = createPopper(triggerRef.current, popupRef.current, {
      placement: getPopperPlacement() as any,
      modifiers,
    });
    setPopperInstance(instance);

    return () => {
      instance.destroy();
      setPopperInstance(null);
    };
  }
}, [actualIsOpen, props.align, props.alignOffset, props.side, props.sideOffset]);

// Handle click events
const handleTriggerClick = (event: React.MouseEvent) => {
  event.stopPropagation();
  if (!props.openOnHover) {
    handleOpenChange(!actualIsOpen, event.nativeEvent, 'click');
  }
};

// Handle hover events
const handleMouseEnter = (event: React.MouseEvent) => {
  if (props.openOnHover) {
    if (hoverTimeoutRef.current) {
      clearTimeout(hoverTimeoutRef.current);
    }
  }
};
```

```
const delay = props.delay ?? 350;
delayTimeoutRef.current = setTimeout(() => {
  handleOpenChange(true, event.nativeEvent, 'hover');
}, delay);
};

const handleMouseLeave = (event: React.MouseEvent) => {
  if (props.openOnHover) {
    if (delayTimeoutRef.current) {
      clearTimeout(delayTimeoutRef.current);
    }

    hoverTimeoutRef.current = setTimeout(() => {
      handleOpenChange(false, event.nativeEvent, 'hover');
    }, 100);
  }
};

// Handle outside clicks
useEffect(() => {
  const handleClickOutside = (event: MouseEvent) => {
    if (
      actualisOpen &&
      triggerRef.current &&
      popupRef.current &&
      !triggerRef.current.contains(event.target as Node) &&
      !popupRef.current.contains(event.target as Node)
    ) {
      handleOpenChange(false, event, 'outside-click');
    }
  };
}

if (actualisOpen) {
  document.addEventListener('mousedown', handleClickOutside);
  return () => document.removeEventListener('mousedown', handleClickOutside);
}
}, [actualisOpen]);

// Handle escape key
useEffect(() => {
  const handleEscape = (event: KeyboardEvent) => {
    if (event.key === 'Escape' && actualisOpen) {
      handleOpenChange(false, event, 'escape');
    }
  };
}

if (actualisOpen) {
  document.addEventListener('keydown', handleEscape);
  return () => document.removeEventListener('keydown', handleEscape);
}
}, [actualisOpen]);
```

```
// Cleanup timeouts
useEffect(() => {
  return () => {
    if (hoverTimeoutRef.current) clearTimeout(hoverTimeoutRef.current);
    if (delayTimeoutRef.current) clearTimeout(delayTimeoutRef.current);
  };
}, []);

const popoverStyles = {
  position: 'relative' as const,
  display: 'inline-block',
};

const triggerStyles = {
  background: 'none',
  border: 'none',
  padding: 0,
  cursor: 'pointer',
  display: 'inline-block',
};

const popupStyles = {
  backgroundColor: '#fff',
  border: '1px solid #e1e5e9',
  borderRadius: '8px',
  boxShadow: '0 4px 12px rgba(0, 0, 0, 0.15)',
  padding: '12px',
  zIndex: 1000,
  maxWidth: '320px',
  fontSize: '14px',
  lineHeight: '1.4',
  visibility: actualIsOpen ? 'visible' : 'hidden',
  opacity: actualIsOpen ? 1 : 0,
  transition: 'opacity 0.2s ease, visibility 0.2s ease',
};

const titleStyles = {
  fontWeight: '600',
  marginBottom: '8px',
  fontSize: '16px',
  color: '#1f2937',
};

const contentStyles = {
  color: '#374151',
};

return (
  <div style={popoverStyles}>
    <div
      ref={triggerRef}
      style={triggerStyles}
      onClick={handleTriggerClick}
      onMouseEnter={handleMouseEnter}>
```

```
        onMouseLeave={handleMouseLeave}
      >
      {target}
    </div>

    {/* Portal-like behavior using fixed positioning */}
    <div
      ref={popupRef}
      style={{
        ...popupStyles,
        position: 'absolute',
        top: 0,
        left: 0,
      }}
      onMouseEnter={() => {
        if (props.openOnHover && hoverTimeoutRef.current) {
          clearTimeout(hoverTimeoutRef.current);
        }
      }}
      onMouseLeave={handleMouseLeave}
    >
      {title && <div style={titleStyles}>{title}</div>}
      <div style={contentStyles}>{defaultContent}</div>
    </div>
  </div>
);

}
```

Task 3

GPT 4.1

Dropdown.tsx

```
import React, { useState, useRef, useEffect } from "react";
import "./Dropdown.css";

export type DropdownOption = {
  label: string;
  value: string;
  disabled?: boolean;
};

type DropdownProps = {
  label?: string;
  placeholder?: string;
  options: DropdownOption[];
  value?: string | string[];
  onChange?: (value: string | string[]) => void;
  multiple?: boolean;
  searchable?: boolean;
};
```

```
condensed?: boolean; // For condensed multi-value display
};

export function Dropdown({
  label,
  placeholder = "Placeholder",
  options,
  value,
  onChange,
  multiple = false,
  searchable = false,
  condensed = false,
}: DropdownProps) {
  const [open, setOpen] = useState(false);
  const [search, setSearch] = useState("");
  const [focused, setFocused] = useState<number | null>(null);
  const ref = useRef<HTMLDivElement>(null);

  // Handle outside click
  useEffect(() => {
    const handler = (e: MouseEvent) => {
      if (!ref.current?.contains(e.target as Node)) setOpen(false);
    };
    if (open) window.addEventListener("mousedown", handler);
    return () => window.removeEventListener("mousedown", handler);
  }, [open]);

  // Filter options for search
  const filteredOptions = searchable
    ? options.filter(
        (opt) =>
          opt.label.toLowerCase().includes(search.toLowerCase()) ||
          opt.value.toLowerCase().includes(search.toLowerCase())
      )
    : options;

  // Selected values
  const selectedValues = multiple
    ? Array.isArray(value) ? value : []
    : typeof value === "string" ? value : "";

  // Click handler for option
  const handleOptionClick = (v: string, disabled?: boolean) => {
    if (disabled) return;
    if (multiple) {
      let arr = Array.isArray(selectedValues) ? [...selectedValues] : [];
      if (arr.includes(v)) {
        arr = arr.filter((val) => val !== v);
      } else {
        arr.push(v);
      }
      onChange?(arr);
    } else {
      onChange?(v);
    }
  };
}
```

```
        setOpen(false);
    }
};

// Remove value (multi)
const removeValue = (v: string) => {
    if (!multiple) return;
    let arr = Array.isArray(selectedValues) ? [...selectedValues] : [];
    arr = arr.filter((val) => val !== v);
    onChange?.(arr);
};

// For condensed display (multi)
const condensedDisplay = () => {
    if (!multiple || !condensed || !Array.isArray(selectedValues)) return null;
    const visible = selectedValues.slice(0, 1);
    const extra = selectedValues.length - 1;
    return (
        <div className="dropdown-tags-condensed">
            <span className="dropdown-tag">
                {options.find((o) => o.value === visible[0])?.label ?? visible[0]}
                <button className="dropdown-tag-remove" onClick={() =>
removeValue(visible[0])}>#10005;</button>
            </span>
            {extra > 0 && <span className="dropdown-tag">+{extra}</span>}
        </div>
    );
};

// Regular multi value tags
const regularTags = () => {
    if (!multiple || condensed || !Array.isArray(selectedValues)) return null;
    return (
        <div className="dropdown-tags">
            {selectedValues.map((v) => (
                <span className="dropdown-tag" key={v}>
                    {options.find((o) => o.value === v)?.label ?? v}
                    <button className="dropdown-tag-remove" onClick={() =>
removeValue(v)}>#10005;</button>
                </span>
            )));
        </div>
    );
};

// Keyboard navigation
const handleKeyDown = (e: React.KeyboardEvent) => {
    if (!open) return;
    const enabledOptions = filteredOptions.filter((opt) => !opt.disabled);
    if (e.key === "ArrowDown") {
        setFocused((prev) =>
            prev === null || prev >= enabledOptions.length - 1 ? 0 : prev + 1
        );
    }
}
```

```
if (e.key === "ArrowUp") {
  setFocused((prev) =>
    prev === null || prev <= 0 ? enabledOptions.length - 1 : prev - 1
  );
}
if (e.key === "Enter" && focused !== null) {
  const opt = enabledOptions[focused];
  if (opt) handleOptionClick(opt.value, opt.disabled);
}
if (e.key === "Escape") setOpen(false);
};

// Render dropdown
return (
  <div className="dropdown-root" ref={ref} tabIndex={0} onKeyDown={handleKeyDown}>
    {label && <div className="dropdown-label">{label}</div>}
    <div
      className={`dropdown-select ${open ? "dropdown-open" : ""} ${
        multiple ? "dropdown-multi" : ""
      }`}
      onClick={() => setOpen((o) => !o)}
    >
      {/* Multi tags */}
      {multiple && condensed ? condensedDisplay() : regularTags()}

      {/* Single value or placeholder */}
      {!multiple && (
        <span className={`dropdown-value ${!selectedValues ? "dropdown-placeholder" : ""}`}>
          {selectedValues
            ? options.find((o) => o.value === selectedValues).label ??
              selectedValues
            : placeholder}
          </span>
      )}
      {/* Multi placeholder */}
      {multiple && !Array.isArray(selectedValues) && (
        <span className="dropdown-placeholder">{placeholder}</span>
      )}

      {/* Search input */}
      {searchable && open && (
        <input
          className="dropdown-search"
          autoFocus
          value={search}
          onChange={(e) => setSearch(e.target.value)}
          placeholder={placeholder}
          onClick={(e) => e.stopPropagation()}
        />
      )}
  </div>
)
```

```
        <span className="dropdown-arrow">➔</span>
    </div>
    {/* Dropdown menu */}
    {open && (
        <div className="dropdown-menu">
            {filteredOptions.map((opt, idx) => {
                const active =
                    multiple
                    ? Array.isArray(selectedValues) &&
selectedValues.includes(opt.value)
                    : selectedValues === opt.value;
                const isFocused = focused === idx;
                return (
                    <div
                        key={opt.value}
                        className={`dropdown-menu-item${active ? " dropdown-active" :
""}${{
                            opt.disabled ? " dropdown-disabled" : ""
                        }}${isFocused ? " dropdown-focused" : ""}`}
                        onClick={() => handleOptionClick(opt.value, opt.disabled)}
                        tabIndex={-1}
                        aria-disabled={!opt.disabled}
                    >
                        {opt.label}
                    </div>
                );
            })}
        </div>
    )}
</div>
}

export default Dropdown;
```

Dropdown.css

```
.dropdown-root {
    display: flex;
    flex-direction: column;
    gap: 7px;
    min-width: 220px;
    font-family: 'Montserrat', Arial, sans-serif;
}

.dropdown-label {
    color: #fff;
    font-size: 15px;
    margin-bottom: 2px;
    font-weight: 500;
}

.dropdown-select {
    background: rgba(18, 26, 56, 0.24);
```

```
border-radius: 7px;
min-height: 36px;
padding: 0 12px;
display: flex;
align-items: center;
cursor: pointer;
position: relative;
border: none;
font-size: 15px;
color: #fff;
transition: background 0.18s;
box-shadow: none;
}
.dropdown-select.dropdown-open {
background: rgba(18, 26, 56, 0.34);
box-shadow: 0 0 0 2px #82a3ff;
}
.dropdown-select.dropdown-multi {
min-height: 40px;
flex-wrap: wrap;
padding-top: 5px;
padding-bottom: 5px;
}
.dropdown-value {
flex: 1;
color: #fff;
font-size: 15px;
}
.dropdown-placeholder {
color: #b2b8cb;
font-size: 15px;
}
.dropdown-arrow {
margin-left: auto;
font-size: 15px;
color: #cf8e3;
pointer-events: none;
}
.dropdown-search {
background: transparent;
border: none;
outline: none;
color: #fff;
font-size: 15px;
padding: 0;
flex: 1;
margin-right: 10px;
}
.dropdown-tags, .dropdown-tags-condensed {
display: flex;
flex-wrap: wrap;
gap: 7px;
align-items: center;
}
```

```
.dropdown-tag {  
    background: rgba(18, 26, 56, 0.38);  
    color: #fff;  
    border-radius: 6px;  
    padding: 2px 10px 2px 10px;  
    font-size: 15px;  
    display: flex;  
    align-items: center;  
    margin-bottom: 2px;  
}  
.dropdown-tag-remove {  
    background: none;  
    border: none;  
    color: #b2b8cb;  
    cursor: pointer;  
    margin-left: 7px;  
    font-size: 15px;  
    padding: 0;  
}  
.dropdown-menu {  
    position: absolute;  
    left: 0;  
    top: 100%;  
    min-width: 220px;  
    background: rgba(18, 26, 56, 0.85);  
    border-radius: 7px;  
    margin-top: 6px;  
    z-index: 20;  
    box-shadow: 0 6px 24px rgba(0, 0, 0, 0.20);  
    padding: 7px 0;  
    display: flex;  
    flex-direction: column;  
    gap: 2px;  
}  
.dropdown-menu-item {  
    padding: 7px 15px;  
    font-size: 15px;  
    color: #fff;  
    cursor: pointer;  
    border-radius: 4px;  
    transition: background 0.14s;  
    background: none;  
    position: relative;  
    outline: none;  
}  
.dropdown-menu-item.dropdown-active,  
.dropdown-menu-item.dropdown-focused {  
    background: rgba(130, 163, 255, 0.45);  
}  
.dropdown-menu-item.dropdown-disabled {  
    color: #b2b8cb;  
    cursor: not-allowed;  
    background: none;  
}
```

```
/* For long value ellipsis */
.dropdown-tag {
  max-width: 160px;
  overflow: hidden;
  text-overflow: ellipsis;
  white-space: nowrap;
}
```

Same.dev

Dropdown.tsx

```
import type React from 'react';
import { useState, useRef, useEffect, useCallback } from 'react';
import { ChevronDown, X } from 'lucide-react';
import type { DropdownProps, DropdownOption, DropdownState } from '../types';

const Dropdown: React.FC<DropdownProps> = ({
  label,
  placeholder = 'Placeholder',
  options,
  value,
  onChange,
  multiple = false,
  searchable = false,
  condensed = false,
  maxVisibleValues = 3,
  disabled = false,
  className = '',
}) => {
  const [state, setState] = useState<DropdownState>({
    isOpen: false,
    searchTerm: '',
    highlightedIndex: -1,
  });

  const dropdownRef = useRef<HTMLDivElement>(null);
  const searchInputRef = useRef<HTMLInputElement>(null);

  // Filter options based on search term
  const filteredOptions = searchable
    ? options.filter(option =>
        option.label.toLowerCase().includes(state.searchTerm.toLowerCase())
      )
    : options;

  // Get current values as array
  const currentValues = multiple
    ? Array.isArray(value) ? value : []
    : Array.isArray(value) ? value : value ? [value] : [];

  return (
    <div>
      <div>
        <input ref={searchInputRef} type="text" value={state.searchTerm} />
        <span>{label}</span>
        <span>{placeholder}</span>
        <span>{options.length}</span>
      </div>
      <div>
        <ul>
          {filteredOptions.map((option, index) =>
            <li key={option.value}>
              <span>{option.label}</span>
              <span>{option.value}</span>
            </li>
          )}
        </ul>
      </div>
    </div>
  );
}
```

```
// Get selected options
const selectedOptions = options.filter(option =>
  currentValues.includes(option.value)
);

// Handle option selection
const handleOptionSelect = useCallback((option: DropdownOption) => {
  if (option.disabled) return;

  if (multiple) {
    const newValues = currentValues.includes(option.value)
      ? currentValues.filter(v => v !== option.value)
      : [...currentValues, option.value];
    onChange(newValues);
  } else {
    console.log(option.value);
    onChange(option.value);
    setState(prev => ({ ...prev, isOpen: false }));
  }
}, [multiple, currentValues, onChange]);

// Handle value removal in multi-select
const handleValueRemove = useCallback((valueToRemove: string) => {
  if (multiple) {
    const newValues = currentValues.filter(v => v !== valueToRemove);
    onChange(newValues);
  }
}, [multiple, currentValues, onChange]);

// Handle dropdown toggle
const handleToggle = useCallback(() => {
  if (disabled) return;
  setState(prev => ({
    ...prev,
    isOpen: !prev.isOpen,
    searchTerm: '',
    highlightedIndex: -1
  }));
}, [disabled]);

// Handle search input change
const handleSearchChange = useCallback((e: React.ChangeEvent<HTMLInputElement>) => {
  setState(prev => ({
    ...prev,
    searchTerm: e.target.value,
    highlightedIndex: -1
  }));
}, []);

// Handle keyboard navigation
const handleKeyDown = useCallback((e: React.KeyboardEvent) => {
  if (!state.isOpen) return;
```

```
switch (e.key) {
  case 'ArrowDown':
    e.preventDefault();
    setState(prev => ({
      ...prev,
      highlightedIndex: Math.min(prev.highlightedIndex + 1,
filteredOptions.length - 1)
    }));
    break;
  case 'ArrowUp':
    e.preventDefault();
    setState(prev => ({
      ...prev,
      highlightedIndex: Math.max(prev.highlightedIndex - 1, 0)
    }));
    break;
  case 'Enter':
    e.preventDefault();
    if (state.highlightedIndex >= 0 && state.highlightedIndex <
filteredOptions.length) {
      handleOptionSelect(filteredOptions[state.highlightedIndex]);
    }
    break;
  case 'Escape':
    setState(prev => ({ ...prev, isOpen: false }));
    break;
}
}, [state.isOpen, state.highlightedIndex, filteredOptions, handleOptionSelect]);

// Close dropdown when clicking outside
useEffect(() => {
  const handleClickOutside = (event: MouseEvent) => {
    if (dropdownRef.current && !dropdownRef.current.contains(event.target as
Node)) {
      setState(prev => ({ ...prev, isOpen: false }));
    }
  };
  document.addEventListener('mousedown', handleClickOutside);
  return () => document.removeEventListener('mousedown', handleClickOutside);
}, []);

// Focus search input when dropdown opens
useEffect(() => {
  if (state.isOpen && searchable && searchInputRef.current) {
    searchInputRef.current.focus();
  }
}, [state.isOpen, searchable]);

// Render display value for single select
const renderSingleValue = () => {
  console.log(currentValues);
  if (currentValues.length === 0) {
```

```
        return <span className="text-slate-400">{placeholder}</span>;
    }
    const selectedOption = options.find(opt => opt.value === currentValues[0]);
    return <span className="text-white">{selectedOption?.label}</span>;
};

// Render value chips for multi-select
const renderMultiValues = () => {
    if (currentValues.length === 0) {
        return <span className="text-slate-400">{placeholder}</span>;
    }

    const visibleValues = condensed ? selectedOptions.slice(0, maxVisibleValues) : selectedOptions;
    const remainingCount = selectedOptions.length - visibleValues.length;

    return (
        <div className="flex flex-wrap gap-1">
            {visibleValues.map((option) => (
                <div
                    key={option.value}
                    className="bg-slate-600 text-white px-2 py-1 rounded text-sm flex items-center gap-1"
                >
                    <span>{option.label}</span>
                    <button
                        onClick={(e) => {
                            e.stopPropagation();
                            handleValueRemove(option.value);
                        }}
                        className="hover:bg-slate-500 rounded-full p-0.5"
                    >
                        <X size={12} />
                    </button>
                </div>
            ))}
            {condensed && remainingCount > 0 && (
                <div className="bg-slate-600 text-white px-2 py-1 rounded text-sm">
                    +{remainingCount}
                </div>
            )}
        </div>
    );
};

return (
    <div className={`relative ${className}`} ref={dropdownRef}>
        {/* Label */}
        <label className="block text-white text-sm font-medium mb-2">
            {label}
        </label>

        {/* Main dropdown trigger */}
        <div
```

```
    onClick={handleToggle}
    onKeyDown={handleKeyDown}
    tabIndex={disabled ? -1 : 0}
    className={`relative bg-slate-700 border border-slate-600 rounded-lg px-3 py-2 cursor-pointer transition-colors min-h-[40px] flex items-center ${disabled ? 'opacity-50 cursor-not-allowed' : 'hover:bg-slate-600'} ${state.isOpen ? 'ring-2 ring-blue-500' : ''}`}
  `}

  >
  <div className="flex-1 flex items-center">
    {multiple ? renderMultiValues() : renderSingleValue()}
  </div>
  <ChevronDown
    size={16}
    className={`text-slate-400 transition-transform ${state.isOpen ? 'rotate-180' : ''}`}
  />
</div>

/* Dropdown menu */
{state.isOpen && (
  <div className="absolute z-50 top-full left-0 right-0 mt-1 bg-slate-700 border border-slate-600 rounded-lg shadow-lg max-h-60 overflow-y-auto">
    /* Search input */
    {searchable && (
      <div className="p-2 border-b border-slate-600">
        <input
          ref={searchInputRef}
          type="text"
          value={state.searchTerm}
          onChange={handleSearchChange}
          placeholder="Search..."
          className="w-full bg-slate-600 text-white placeholder-slate-400 px-3 py-1 rounded border-0 focus:outline-none focus:ring-1 focus:ring-blue-500"
        />
      </div>
    )}
    /* Multi-select values display */
    {multiple && searchable && currentValues.length > 0 && (
      <div className="p-2 border-b border-slate-600">
        <div className="flex flex-wrap gap-1">
          {selectedOptions.map((option) => (
            <div
              key={option.value}
              className="bg-slate-600 text-white px-2 py-1 rounded text-sm flex items-center gap-1"
            >
              <span>{option.label}</span>
              <button
                onClick={(e) => {
                  e.stopPropagation();
                  handleValueRemove(option.value);
                }}
              >
                <span>X</span>
              </button>
            </div>
          ))}
        </div>
      </div>
    )}
  </div>
)}>
```

```
        })}
        className="hover:bg-slate-500 rounded-full p-0.5"
      >
    <X size={10} />
  </button>
</div>
))})
</div>
</div>
)})

{/* Options list */}
<div className="py-1">
  {filteredOptions.length === 0 ? (
    <div className="px-3 py-2 text-slate-400 text-sm">No options
found</div>
  ) : (
    filteredOptions.map((option, index) => {
      const isSelected = currentValues.includes(option.value);
      const isHighlighted = index === state.highlightedIndex;

      return (
        <div
          key={option.id}
          onClick={() => handleOptionSelect(option)}
          className={`px-3 py-2 cursor-pointer transition-colors text-sm
${option.disabled
  ? 'text-slate-500 cursor-not-allowed'
  : 'text-white hover:bg-blue-600'
}
${isSelected ? 'bg-blue-600' : ''}
${isHighlighted ? 'bg-blue-600' : ''}
`}
        >
          {option.label}
        </div>
      );
    })
  )
</div>
</div>
)};

export default Dropdown;
```

Cursor

Dropdown.tsx

```
import React, { useState } from 'react';
import './Dropdown.css';

type Item = {
  label: string;
  value: string;
  disabled?: boolean;
};

type DropdownProps = {
  label: string;
  items: Item[];
  multiSelect?: boolean;
};

const Dropdown: React.FC<DropdownProps> = ({ label, items, multiSelect }) => {
  const [selectedItems, setSelectedItems] = useState<Item[]>([]);
  const [isOpen, setIsOpen] = useState(false);

  const toggleDropdown = () => setIsOpen(!isOpen);

  const handleSelect = (item: Item) => {
    if (multiSelect) {
      if (selectedItems.includes(item)) {
        setSelectedItems(selectedItems.filter(i => i !== item));
      } else {
        setSelectedItems([...selectedItems, item]);
      }
    } else {
      setSelectedItems([item]);
      setIsOpen(false);
    }
  };
}

return (
  <div className="dropdown">
    <label>{label}</label>
    <div className="dropdown-toggle" onClick={toggleDropdown}>
      {selectedItems.length > 0 ? selectedItems.map(i => i.label).join(', ') :
      'Select...'}
    </div>
    {isOpen && (
      <div className="dropdown-menu">
        {items.map(item => (
          <div
            key={item.value}
            className={`dropdown-item ${item.disabled ? 'disabled' : ''}`}
            onClick={() => !item.disabled && handleSelect(item)}
          >
            {item.label}
          </div>
        )));
    )}
  </div>
)
```

```
        </div>
    )}
</div>
);
};

export default Dropdown;
```

Dropdown.css

```
.dropdown {
  position: relative;
  width: 200px;
}

.dropdown-toggle {
  padding: 10px;
  border: 1px solid #ccc;
  cursor: pointer;
  background-color: #f9f9f9;
}

.dropdown-menu {
  position: absolute;
  top: 100%;
  left: 0;
  right: 0;
  border: 1px solid #ccc;
  background-color: white;
  z-index: 1000;
}

.dropdown-item {
  padding: 10px;
  cursor: pointer;
}

.dropdown-item.disabled {
  color: gray;
  cursor: not-allowed;
}

.dropdown-item:hover:not(.disabled) {
  background-color: #f0f0f0;
}
```

Claude Opus 4

Dropdown.tsx

```
import React, { useState, useRef, useEffect } from 'react';

// Types
type DropdownMode = 'single' | 'search' | 'multi';
type DropdownItem = {
  id: string;
  label: string;
  disabled?: boolean;
};

type DropdownProps = {
  label?: string;
  placeholder?: string;
  items: DropdownItem[];
  mode?: DropdownMode;
  value?: string | string[];
  onChange?: (value: string | string[]) => void;
  searchable?: boolean;
  condensed?: boolean;
};

const DropdownComponent: React.FC<DropdownProps> = ({label,
placeholder = 'Placeholder',
items,
mode = 'single',
value,
onChange,
searchable = false,
condensed = false,
}) => {
  const [isOpen, setIsOpen] = useState(false);
  const [searchQuery, setSearchQuery] = useState('');
  const [selectedValues, setSelectedValues] = useState<string[]>(() => {
    if (mode === 'multi' && Array.isArray(value)) return value;
    if (mode === 'single' && typeof value === 'string') return [value];
    return [];
  });
  const dropdownRef = useRef<HTMLDivElement>(null);

  // Handle click outside
  useEffect(() => {
    const handleClickOutside = (event: MouseEvent) => {
      if (dropdownRef.current && !dropdownRef.current.contains(event.target as Node)) {
        setIsOpen(false);
      }
    };
    document.addEventListener('mousedown', handleClickOutside);
    return () => document.removeEventListener('mousedown', handleClickOutside);
  }, []);
}
```

```
// Update selected values when props change
useEffect(() => {
  if (mode === 'multi' && Array.isArray(value)) {
    setSelectedValues(value);
  } else if (mode === 'single' && typeof value === 'string') {
    setSelectedValues([value]);
  }
}, [value, mode]);

const filteredItems = items.filter(item =>
  item.label.toLowerCase().includes(searchQuery.toLowerCase())
);

const handleItemClick = (itemId: string) => {
  const item = items.find(i => i.id === itemId);
  if (item?.disabled) return;

  if (mode === 'single' || mode === 'search') {
    setSelectedValues([itemId]);
    onChange?(itemId);
    setIsOpen(false);
    setSearchQuery('');
  } else if (mode === 'multi') {
    const newValues = selectedValues.includes(itemId)
      ? selectedValues.filter(v => v !== itemId)
      : [...selectedValues, itemId];
    setSelectedValues(newValues);
    onChange?(newValues);
  }
};

const removeValue = (itemId: string, e: React.MouseEvent) => {
  e.stopPropagation();
  const newValues = selectedValues.filter(v => v !== itemId);
  setSelectedValues(newValues);
  onChange?(mode === 'multi' ? newValues : '');
};

const getDisplayText = () => {
  if (selectedValues.length === 0) return placeholder;

  const selectedItems = items.filter(item => selectedValues.includes(item.id));

  if (mode === 'single' || mode === 'search') {
    return selectedItems[0] ?.label || placeholder;
  }

  if (mode === 'multi' && condensed && selectedValues.length > 1) {
    return `${selectedItems[0] ?.label} + ${selectedValues.length - 1}`;
  }

  return placeholder;
};
```

```
const showSearchInput = mode === 'search' && isOpen;

return (
  <div ref={dropdownRef} style={{ position: 'relative', width: '280px' }}>
    {label && (
      <div style={{
        fontSize: '14px',
        fontWeight: '600',
        color: '#e0e0e0',
        marginBottom: '8px',
      }}>
        {label}
      </div>
    )}
  <div
    onClick={() => setIsOpen(!isOpen)}
    style={{
      backgroundColor: '#2a3b5c',
      border: '1px solid #3a4b6c',
      borderRadius: '8px',
      padding: '10px 40px 10px 16px',
      cursor: 'pointer',
      position: 'relative',
      minHeight: '44px',
      display: 'flex',
      alignItems: 'center',
      flexWrap: mode === 'multi' && !condensed ? 'wrap' : 'nowrap',
      gap: '8px',
    }}
  >
    {mode === 'multi' && selectedValues.length > 0 && !condensed ? (
      <>
        {selectedValues.map(val => {
          const item = items.find(i => i.id === val);
          return (
            <div
              key={val}
              style={{
                backgroundColor: '#3a4b6c',
                borderRadius: '20px',
                padding: '4px 12px',
                fontSize: '14px',
                color: '#e0e0e0',
                display: 'inline-flex',
                alignItems: 'center',
                gap: '6px',
                maxWidth: '150px',
              }}
            >
              <span style={{
                overflow: 'hidden',
                textOverflow: 'ellipsis',
                whiteSpace: 'nowrap',
              }}>
                {item.name}
              </span>
            </div>
          )
        )}
      </>
    ) : (
      <div style={{
        height: '44px',
        width: '100%',
        background: '#3a4b6c',
        border: '1px solid #3a4b6c',
        borderRadius: '8px',
        padding: '10px 40px 10px 16px',
        display: 'flex',
        alignItems: 'center',
        gap: '8px',
      }}>
        {label}
        {placeholder}
      </div>
    )}
  </div>
)
```

```
        }
        {item?.label}
    </span>
    <button
        onClick={(e) => removeValue(val, e)}
        style={{
            background: 'none',
            border: 'none',
            color: '#8090b0',
            cursor: 'pointer',
            padding: '0',
            fontSize: '16px',
            lineHeight: '1',
        }}
    >
        ×
    </button>
</div>
);
})}
</>
) : (
    <span style={{ color: selectedValues.length > 0 ? '#e0e0e0' : '#8090b0' }}>
        {getdisplayText()}
    </span>
)
}

<svg
    style={{
        position: 'absolute',
        right: '16px',
        top: '50%',
        transform: `translateY(-50%) rotate(${isOpen ? '180deg' : '0deg'})`,
        transition: 'transform 0.2s',
        width: '20px',
        height: '20px',
    }}
    viewBox="0 0 20 20"
    fill="#8090b0"
>
    <path d="M5.293 7.293a1 1 0 011.414 0L10 10.586l3.293-3.293a1 1 0 111.414 1.414l-4 4a1 1 0 01-1.414 0L-4-4a1 1 0 010-1.414z" />
</svg>
</div>

{isOpen && (
    <div style={{
        position: 'absolute',
        top: 'calc(100% + 8px)',
        left: '0',
        right: '0',
        backgroundColor: '#1a2a4a',
        border: '1px solid #3a4b6c',
    }}>
```

```
borderRadius: '8px',
boxShadow: '0 4px 12px rgba(0, 0, 0, 0.3)',
zIndex: 1000,
maxHeight: '300px',
overflowY: 'auto',
})}>
{showSearchInput && (
  <div style={{ padding: '8px' }}>
    <input
      type="text"
      value={searchQuery}
      onChange={(e) => setSearchQuery(e.target.value)}
      placeholder="Search..."
      autoFocus
      style={{
        width: '100%',
        padding: '8px 12px',
        backgroundColor: '#2a3b5c',
        border: '1px solid #3a4b6c',
        borderRadius: '6px',
        color: '#e0e0e0',
        outline: 'none',
      }}
    />
  </div>
)}

<div style={{ padding: '4px' }}>
{filteredItems.map((item) => {
  const isSelected = selectedValues.includes(item.id);
  const isEnabled = item.disabled;

  return (
    <div
      key={item.id}
      onClick={() => handleItemClick(item.id)}
      style={{
        padding: '10px 16px',
        cursor: isEnabled ? 'not-allowed' : 'pointer',
        backgroundColor: isSelected ? '#3a5b8c' : 'transparent',
        color: isEnabled ? '#506080' : '#e0e0e0',
        borderRadius: '6px',
        margin: '2px 0',
        transition: 'background-color 0.15s',
        position: 'relative',
        paddingRight: isSelected && mode === 'multi' ? '40px' :
      '16px',
      }}
      onMouseEnter={(e) => {
        if (!isEnabled && !isSelected) {
          e.currentTarget.style.backgroundColor = '#2a3b5c';
        }
      }}
      onMouseLeave={(e) => {
```

```
        if (!isDisabled && !isSelected) {
            e.currentTarget.style.backgroundColor = 'transparent';
        }
    })
}

>
{item.label}
{isSelected && (mode === 'multi' || mode === 'search') && (
    <svg
        style={{
            position: 'absolute',
            right: '16px',
            top: '50%',
            transform: 'translateY(-50%)',
            width: '20px',
            height: '20px',
        }}
        viewBox="0 0 20 20"
        fill="#7090d0"
    >
        <path d="M16.707 5.293a1 1 0 010 1.414l-8 8a1 1 0 01-1.414
01-4-4a1 1 0 011.414-1.414L8 12.586l7.293-7.293a1 1 0 011.414 0z" />
    </svg>
)
)
</div>
);
})}
</div>
</div>
)
)
</div>
);
};
```

Claude Sonnet 4

Dropdown.tsx

```
import React, { useState, useRef, useEffect } from 'react';

// Types
type DropdownItem = {
  id: string;
  label: string;
  disabled?: boolean;
};

type DropdownProps = {
  label: string;
  placeholder?: string;
  items: DropdownItem[];
  value?: string | string[];
};
```

```
onChange?: (value: string | string[]) => void;
multiSelect?: boolean;
searchable?: boolean;
displayMode?: 'regular' | 'condensed'; // For multi-select display
disabled?: boolean;
};

const Dropdown: React.FC<DropdownProps> = ({
  label,
  placeholder = "Placeholder",
  items,
  value,
  onChange,
  multiSelect = false,
  searchable = false,
  displayMode = 'regular',
  disabled = false
}) => {
  const [isOpen, setIsOpen] = useState(false);
  const [searchTerm, setSearchTerm] = useState('');
  const dropdownRef = useRef<HTMLDivElement>(null);
  const searchInputRef = useRef<HTMLInputElement>(null);

  // Handle click outside
  useEffect(() => {
    const handleClickOutside = (event: MouseEvent) => {
      if (dropdownRef.current && !dropdownRef.current.contains(event.target as Node)) {
        setIsOpen(false);
        setSearchTerm('');
      }
    };

    document.addEventListener('mousedown', handleClickOutside);
    return () => document.removeEventListener('mousedown', handleClickOutside);
  }, []);

  // Focus search input when dropdown opens
  useEffect(() => {
    if (isOpen && searchable && searchInputRef.current) {
      searchInputRef.current.focus();
    }
  }, [isOpen, searchable]);

  // Filter items based on search
  const filteredItems = items.filter(item =>
    item.label.toLowerCase().includes(searchTerm.toLowerCase())
  );

  // Get selected items
  const selectedItems = items.filter(item => {
    if (Array.isArray(value)) {
      return value.includes(item.id);
    }
  });
}
```

```
        return value === item.id;
    });

// Handle item selection
const handleItemSelect = (itemId: string) => {
    if (multiSelect) {
        const currentValue = Array.isArray(value) ? value : [];
        const newValue = currentValue.includes(itemId)
            ? currentValue.filter(id => id !== itemId)
            : [...currentValue, itemId];
        onChange?.(newValue);
    } else {
        onChange?.(itemId);
        setIsOpen(false);
        setSearchTerm('');
    }
};

// Remove selected item (for multi-select)
const handleRemoveItem = (itemId: string, event: React.MouseEvent) => {
    event.stopPropagation();
    if (Array.isArray(value)) {
        const newValue = value.filter(id => id !== itemId);
        onChange?.(newValue);
    }
};

// Get display text for trigger
const getDisplayText = () => {
    if (!selectedItems.length) return placeholder;

    if (!multiSelect) {
        return selectedItems[0] ?.label || placeholder;
    }

    if (displayMode === 'condensed' && selectedItems.length > 1) {
        return `${selectedItems[0].label} +${selectedItems.length - 1}`;
    }

    return selectedItems.map(item => item.label).join(', ');
};

// Render selected value tags (for multi-select regular mode)
const renderValueTags = () => {
    if (!multiSelect || displayMode === 'condensed') return null;

    return (
        <div style={{ display: 'flex', flexWrap: 'wrap', gap: '4px', marginTop: '8px' }}>
            {selectedItems.map(item => (
                <div
                    key={item.id}
                    style={{
                        display: 'flex',
                        // ...other styles
                    }}
            )};
    );
};
```

```
        alignItems: 'center',
        gap: '4px',
        backgroundColor: 'rgba(255, 255, 255, 0.2)',
        padding: '4px 8px',
        borderRadius: '16px',
        fontSize: '12px',
        color: '#fff'
    )}
>
<span>{item.label}</span>
<button
    onClick={(e) => handleRemoveItem(item.id, e)}
    style={{
        background: 'none',
        border: 'none',
        color: '#fff',
        cursor: 'pointer',
        padding: '0',
        display: 'flex',
        alignItems: 'center',
        fontSize: '14px'
    }}
>
    X
</button>
</div>
))})
</div>
);
};

// Base styles
const containerStyle: React.CSSProperties = {
    position: 'relative',
    width: '220px',
    fontFamily: 'system-ui, -apple-system, sans-serif'
};

const labelStyle: React.CSSProperties = {
    display: 'block',
    fontSize: '14px',
    fontWeight: '500',
    color: '#fff',
    marginBottom: '8px'
};

const triggerStyle: React.CSSProperties = {
    width: '100%',
    padding: '8px 12px',
    backgroundColor: 'rgba(255, 255, 255, 0.1)',
    border: '1px solid rgba(255, 255, 255, 0.2)',
    borderRadius: '6px',
    color: '#fff',
    cursor: disabled ? 'not-allowed' : 'pointer',
}
```

```
display: 'flex',
justifyContent: 'space-between',
alignItems: 'center',
fontSize: '14px',
opacity: disabled ? 0.5 : 1,
minHeight: '36px'
};

const dropdownStyle: React.CSSProperties = {
position: 'absolute',
top: '100%',
left: '0',
right: '0',
backgroundColor: 'rgba(30, 41, 59, 0.95)',
border: '1px solid rgba(255, 255, 255, 0.2)',
borderRadius: '6px',
marginTop: '4px',
zIndex: 1000,
maxHeight: '200px',
overflowY: 'auto'
};

const searchInputStyle: React.CSSProperties = {
width: '100%',
padding: '8px 12px',
backgroundColor: 'transparent',
border: 'none',
borderBottom: '1px solid rgba(255, 255, 255, 0.2)',
color: '#fff',
fontSize: '14px',
outline: 'none'
};

const itemStyle = (item: DropdownItem, isSelected: boolean): React.CSSProperties
=> ({
padding: '8px 12px',
cursor: item.disabled ? 'not-allowed' : 'pointer',
backgroundColor: isSelected ? 'rgba(99, 102, 241, 0.8)' : 'transparent',
color: item.disabled ? 'rgba(255, 255, 255, 0.4)' : '#fff',
fontSize: '14px',
borderBottom: '1px solid rgba(255, 255, 255, 0.1)',
display: 'flex',
alignItems: 'center',
gap: '8px'
});

return (
<div ref={dropdownRef} style={containerStyle}>
  <label style={labelStyle}>{label}</label>

  <div
    onClick={() => !disabled && setIsOpen(!isOpen)}
    style={triggerStyle}
  >
```

```
<span style={{
  overflow: 'hidden',
  textOverflow: 'ellipsis',
  whiteSpace: 'nowrap',
  flex: 1
}}>
  {getdisplayText()}
</span>
<span style={{
  marginLeft: '8px',
  transform: isOpen ? 'rotate(180deg)' : 'rotate(0deg)',
  transition: 'transform 0.2s ease'
}}>
  ▼
</span>
</div>

{renderValueTags()}

{isOpen && (
  <div style={dropdownStyle}>
    {searchable && (
      <input
        ref={searchInputRef}
        type="text"
        placeholder="Search..."
        value={searchTerm}
        onChange={(e) => setSearchTerm(e.target.value)}
        style={searchInputStyle}
      />
    )}
    {filteredItems.map((item, index) => {
      const isSelected = Array.isArray(value)
        ? value.includes(item.id)
        : value === item.id;

      return (
        <div
          key={item.id}
          onClick={() => !item.disabled && handleItemSelect(item.id)}
          style={{
            ...itemStyle(item, isSelected),
            borderBottom: index === filteredItems.length - 1 ? 'none' : '1px
solid rgba(255, 255, 255, 0.1)'
          }}
        >
          {multiSelect && (
            <input
              type="checkbox"
              checked={isSelected}
              onChange={() => {}}
              style={{ margin: 0 }}
            />
          )}
        
```

```
        )}
      <span>{item.label}</span>
    </div>
  );
})}

{filteredItems.length === 0 && (
  <div style={{ padding: '12px', color: 'rgba(255, 255, 255, 0.6)',  
fontSize: '14px' }}>
    No items found
  </div>
)
</div>
)
</div>
);
};
```

Task 4

GPT 4.1

ProductPage.tsx

```
import React, { useEffect, useState } from "react";

type Banner = {
  id: string;
  message: string;
  style: "primary" | "warning" | "success";
};

type Product = { [key: string]: any };

const BANNER_STYLE: Record<Banner["style"], React.CSSProperties> = {
  primary: {
    background: "#82a3ff",
    color: "#fff",
    padding: "12px 20px",
    borderRadius: "8px",
    marginBottom: "10px",
    fontWeight: 500,
  },
  warning: {
    background: "#b71818",
    color: "#fff",
    padding: "12px 20px",
    borderRadius: "8px",
    marginBottom: "10px",
    fontWeight: 500,
  },
};
```

```
success: {
  background: "#3bd671",
  color: "#fff",
  padding: "12px 20px",
  borderRadius: "8px",
  marginBottom: "10px",
  fontWeight: 500,
},
};

export default function ProductPage() {
  const [banners, setBanners] = useState<Banner[]>([]);
  const [products, setProducts] = useState<Product[]>([]);
  const [loadingProducts, setLoadingProducts] = useState(true);
  const [loadingBanners, setLoadingBanners] = useState(true);

  // Fetch banners
  useEffect(() => {
    setLoadingBanners(true);
    fetch("http://localhost:8080/rest/banners")
      .then((res) => res.json())
      .then((data) => {
        // Handle array or single banner object
        if (Array.isArray(data)) {
          setBanners(data);
        } else if (data && typeof data === "object") {
          setBanners([data]);
        } else {
          setBanners([]);
        }
      })
      .catch(() => setBanners([]))
      .finally(() => setLoadingBanners(false));
  }, []);

  // Fetch products
  useEffect(() => {
    setLoadingProducts(true);
    fetch("http://localhost:8080/rest/product/search", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({
        includeProductOriginData: false,
        includeSystemBulkUploads: false,
      }),
    })
      .then((res) => res.json())
      .then((data) => {
        if (data && Array.isArray(data.data)) {
          setProducts(data.data);
        } else {
          setProducts([]);
        }
      })
  });
}
```

```
.catch(() => setProducts([]))
    .finally(() => setLoadingProducts(false));
}, []);

return (
  <div style={{ maxWidth: 900, margin: "40px auto", fontFamily: "Montserrat, Arial, sans-serif" }}>
    <h1 style={{ fontWeight: 700, fontSize: 28, marginBottom: 30, color: "#223" }}>
      Product Dashboard
    </h1>

    {/* Banners */}
    {loadingBanners ? (
      <div style={{ marginBottom: 12, color: "#888" }}>Loading banners...</div>
    ) : (
      banners.map((banner) => (
        <div key={banner.id} style={BANNER_STYLE[banner.style]}>
          {banner.message}
        </div>
      ))
    )}
  
```

{/* Products */}

```
<h2 style={{ fontWeight: 600, fontSize: 20, margin: "22px 0 12px", color: "#223" }}>
  Products
</h2>
{loadingProducts ? (
  <div style={{ color: "#888", fontSize: 16 }}>Loading products...</div>
) : products.length === 0 ? (
  <div style={{ color: "#b71818", fontSize: 16 }}>No products found.</div>
) : (
  <table
    style={{
      width: "100%",
      borderCollapse: "collapse",
      background: "#fff",
      borderRadius: "10px",
      overflow: "hidden",
      boxShadow: "0 4px 22px rgba(18,26,56,0.12)",
    }}
  >
    <thead>
      <tr>
        {Object.keys(products[0]).map((key) => (
          <th
            key={key}
            style={{
              textAlign: "left",
              background: "#f6f7fc",
              color: "#223",
              padding: "10px 14px",
              borderBottom: "1px solid #ececfc",
            }}
          >
        ))}
      </tr>
    </thead>
    <tbody>
      {products.map((product) => (
        <tr>
          {Object.keys(product).map((key) => (
            <td
              key={key}
              style={{
                width: "150px",
                verticalAlign: "top",
                padding: "10px 14px",
              }}
            >
              {product[key]}
            </td>
          ))}
        </tr>
      ))}
    </tbody>
  </table>
)
}

```

```
        fontWeight: 600,
    )}
>
    {key}
</th>
))}
</tr>
</thead>
<tbody>
{products.map((prod, idx) => (
    <tr key={idx}>
        {Object.keys(products[0]).map((key) => (
            <td
                key={key}
                style={{
                    padding: "10px 14px",
                    borderBottom: "1px solid #f2f2f2",
                    color: "#223",
                    fontSize: 15,
                }}
            >
                {typeof prod[key] === "object"
                    ? JSON.stringify(prod[key])
                    : String(prod[key])}
            </td>
        )));
    </tr>
))}
</tbody>
</table>
)}
</div>
);
}
```

Same.dev

components/ui/alert.tsx

```
import * as React from "react"
import { cva, type VariantProps } from "class-variance-authority"

import { cn } from "@/lib/utils"

const alertVariants = cva(
    "relative w-full rounded-lg border px-4 py-3 text-sm [&>svg+div]:translate-y-[-3px] [&>svg]:absolute [&>svg]:left-4 [&>svg]:top-4 [&>svg]:text-foreground [&>svg~*]:pl-7",
{
    variants: {
        variant: {

```

```
    default: "bg-background text-foreground",
    destructive:
      "border-destructive/50 text-destructive dark:border-destructive
[&>svg]:text-destructive",
    },
  },
  defaultVariants: {
    variant: "default",
  },
}
)

const Alert = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement> & VariantProps<typeof alertVariants>
>(({ className, variant, ...props }, ref) => (
  <div
    ref={ref}
    role="alert"
    className={cn(alertVariants({ variant }), className)}
    {...props}
  />
))
Alert.displayName = "Alert"

const AlertTitle = React.forwardRef<
  HTMLParagraphElement,
  React.HTMLAttributes<HTMLHeadingElement>
>(({ className, ...props }, ref) => (
  <h5
    ref={ref}
    className={cn("mb-1 font-medium leading-none tracking-tight", className)}
    {...props}
  />
))
AlertTitle.displayName = "AlertTitle"

const AlertDescription = React.forwardRef<
  HTMLParagraphElement,
  React.HTMLAttributes<HTMLParagraphElement>
>(({ className, ...props }, ref) => (
  <div
    ref={ref}
    className={cn("text-sm [&_p]:leading-relaxed", className)}
    {...props}
  />
))
AlertDescription.displayName = "AlertDescription"

export { Alert, AlertTitle, AlertDescription }
```

```
import type * as React from "react"
import { cva, type VariantProps } from "class-variance-authority"

import { cn } from "@/lib/utils"

const badgeVariants = cva(
  "inline-flex items-center rounded-md border px-2.5 py-0.5 text-xs font-semibold
transition-colors focus:outline-none focus:ring-2 focus:ring-ring focus:ring-
offset-2",
{
  variants: {
    variant: {
      default:
        "border-transparent bg-primary text-primary-foreground shadow hover:bg-
primary/80",
      secondary:
        "border-transparent bg-secondary text-secondary-foreground hover:bg-
secondary/80",
      destructive:
        "border-transparent bg-destructive text-destructive-foreground shadow
hover:bg-destructive/80",
      outline: "text-foreground",
    },
  },
  defaultVariants: {
    variant: "default",
  },
}
)

export interface BadgeProps
  extends React.HTMLAttributes<HTMLDivElement>,
  VariantProps<typeof badgeVariants> {}

function Badge({ className, variant, ...props }: BadgeProps) {
  return (
    <div className={cn(badgeVariants({ variant }), className)} {...props} />
  )
}

export { Badge, badgeVariants }
```

components/ui/button.tsx

```
import * as React from "react"
import { Slot } from "@radix-ui/react-slot"
import { cva, type VariantProps } from "class-variance-authority"

import { cn } from "@/lib/utils"

const buttonVariants = cva(
```

```
"inline-flex items-center justify-center gap-2 whitespace nowrap rounded-md
text-sm font-medium transition-colors focus-visible:outline-none focus-
visible:ring-1 focus-visible:ring-ring disabled:pointer-events-none
disabled:opacity-50 [&_svg]:pointer-events-none [&_svg]:size-4 [&_svg]:shrink-0",
{
  variants: {
    variant: {
      default:
        "bg-primary text-primary-foreground shadow hover:bg-primary/90",
      destructive:
        "bg-destructive text-destructive-foreground shadow-sm hover:bg-
destructive/90",
      outline:
        "border border-input bg-background shadow-sm hover:bg-accent hover:text-
accent-foreground",
      secondary:
        "bg-secondary text-secondary-foreground shadow-sm hover:bg-
secondary/80",
      ghost: "hover:bg-accent hover:text-accent-foreground",
      link: "text-primary underline-offset-4 hover:underline",
    },
    size: {
      default: "h-9 px-4 py-2",
      sm: "h-8 rounded-md px-3 text-xs",
      lg: "h-10 rounded-md px-8",
      icon: "h-9 w-9",
    },
  },
  defaultVariants: {
    variant: "default",
    size: "default",
  },
}
)

export interface ButtonProps
  extends React.ButtonHTMLAttributes<HTMLButtonElement>,
  VariantProps<typeof buttonVariants> {
  asChild?: boolean
}

const Button = React.forwardRef<HTMLButtonElement, ButtonProps>(
({ className, variant, size, asChild = false, ...props }, ref) => {
  const Comp = asChild ? Slot : "button"
  return (
    <Comp
      className={cn(buttonVariants({ variant, size, className }))} ref={ref} {...props}>
    />
  )
}
)
Button.displayName = "Button"
```

```
export { Button, buttonVariants }
```

components/ui/card.tsx

```
import * as React from "react"

import { cn } from "@/lib/utils"

const Card = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => (
  <div
    ref={ref}
    className={cn(
      "rounded-xl border bg-card text-card-foreground shadow",
      className
    )}
    {...props}
  />
))
Card.displayName = "Card"

const CardHeader = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => (
  <div
    ref={ref}
    className={cn("flex flex-col space-y-1.5 p-6", className)}
    {...props}
  />
))
CardHeader.displayName = "CardHeader"

const CardTitle = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => (
  <div
    ref={ref}
    className={cn("font-semibold leading-none tracking-tight", className)}
    {...props}
  />
))
CardTitle.displayName = "CardTitle"

const CardDescription = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => (
```

```
<div
  ref={ref}
  className={cn("text-sm text-muted-foreground", className)}
  {...props}
/>
))
CardDescription.displayName = "CardDescription"

const CardContent = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => (
  <div ref={ref} className={cn("p-6 pt-0", className)} {...props} />
))
CardContent.displayName = "CardContent"

const CardFooter = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => (
  <div
    ref={ref}
    className={cn("flex items-center p-6 pt-0", className)}
    {...props}
  />
))
CardFooter.displayName = "CardFooter"

export { Card, CardHeader, CardFooter, CardTitle, CardDescription, CardContent }
```

components/BannerDisplay.tsx

```
import { Alert, AlertDescription } from '@/components/ui/alert';
import { Badge } from '@/components/ui/badge';
import type { Banner } from '@/types/api';
import { AlertCircle, CheckCircle, Info } from 'lucide-react';

interface BannerDisplayProps {
  banners: Banner[];
}

const getBannerIcon = (style: Banner['style']) => {
  switch (style) {
    case 'warning':
      return <AlertCircle className="h-4 w-4" />;
    case 'success':
      return <CheckCircle className="h-4 w-4" />;
    default:
      return <Info className="h-4 w-4" />;
  }
};
```

```
const getBannerVariant = (style: Banner['style']) => {
  switch (style) {
    case 'warning':
      return 'destructive' as const;
    default:
      return 'default' as const;
  }
};

const getBadgeVariant = (style: Banner['style']) => {
  switch (style) {
    case 'warning':
      return 'destructive' as const;
    case 'success':
      return 'default' as const;
    default:
      return 'secondary' as const;
  }
};

export function BannerDisplay({ banners }: BannerDisplayProps) {
  if (banners.length === 0) {
    return null;
  }

  return (
    <div className="space-y-4 mb-6">
      <h2 className="text-2xl font-bold">System Banners</h2>
      <div className="space-y-3">
        {banners.map((banner) => (
          <Alert key={banner.id} variant={getBannerVariant(banner.style)}>
            {getBannerIcon(banner.style)}
            <div className="flex items-start gap-3">
              <AlertDescription className="flex items-center justify-between">
                <span>{banner.message}</span>
                <Badge variant={getBadgeVariant(banner.style)} className="ml-3">
                  {banner.style}
                </Badge>
              </AlertDescription>
            </div>
          </Alert>
        )));
      </div>
    </div>
  );
}
```

components/ProductDisplay.tsx

```
import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card';
import { Badge } from '@/components/ui/badge';
```

```
import type { ProductSearchResponse } from '@/types/api';

interface ProductDisplayProps {
  productData: ProductSearchResponse;
}

export function ProductDisplay({ productData }: ProductDisplayProps) {
  const { data, isLoading } = productData;

  if (isLoading) {
    return (
      <Card>
        <CardHeader>
          <CardTitle>Products</CardTitle>
        </CardHeader>
        <CardContent>
          <div className="flex items-center justify-center py-8">
            <div className="animate-spin rounded-full h-8 w-8 border-b-2 border-gray-900" />
            <span className="ml-2">Loading products...</span>
          </div>
        </CardContent>
      </Card>
    );
  }

  if (!products || products.length === 0) {
    return (
      <Card>
        <CardHeader>
          <CardTitle>Products</CardTitle>
        </CardHeader>
        <CardContent>
          <p className="text-gray-500 text-center py-8">No products found.</p>
        </CardContent>
      </Card>
    );
  }

  // Get all unique keys from all products to create table headers
  const allKeys = Array.from(
    new Set(products.flatMap(product => Object.keys(product)))
  );

  const renderValue = (value: any): React.ReactNode => {
    if (value === null || value === undefined) {
      return <span className="text-gray-400">--</span>;
    }

    if (typeof value === 'boolean') {
      return (
        <Badge variant={value ? 'default' : 'secondary'}>
          {value ? 'Yes' : 'No'}
        </Badge>
      );
    }
  }
}
```

```
        );
    }

    if (typeof value === 'object') {
        return <span className="text-xs text-gray-600">{JSON.stringify(value)}
    }
}

return String(value);
};

return (
<Card>
<CardHeader>
    <CardTitle className="flex items-center justify-between">
        Products
        <Badge variant="outline">{products.length} items</Badge>
    </CardTitle>
</CardHeader>
<CardContent>
    <div className="overflow-x-auto">
        <table className="w-full border-collapse">
            <thead>
                <tr className="border-b">
                    {allKeys.map((key) => (
                        <th
                            key={key}
                            className="text-left p-3 font-medium text-gray-700 bg-gray-50"
                        >
                            {key}
                        </th>
                    )))
                </tr>
            </thead>
            <tbody>
                {products.map((product, index) => (
                    <tr key={`product-${index}-${JSON.stringify(product).slice(0, 20)}`} className="border-b hover:bg-gray-50">
                        {allKeys.map((key) => (
                            <td key={`${index}-${key}`} className="p-3">
                                {renderValue(product[key])}
                            </td>
                        )))
                    </tr>
                )))
            </tbody>
        </table>
    </div>
</CardContent>
</Card>
);
}
```

lib/utils.ts

```
import { clsx, type ClassValue } from "clsx";
import { twMerge } from "tailwind-merge";

export function cn(...inputs: ClassValue[]) {
  return twMerge(clsx(inputs));
}
```

services/api.ts

```
import type { Banner, ProductSearchResponse, ProductSearchRequest } from
'@/types/api';

const BASE_URL = 'http://localhost:8080';

export const api = {
  async fetchBanners(): Promise<Banner[]> {
    try {
      const response = await fetch(`#${BASE_URL}/rest/banners`);
      if (!response.ok) {
        throw new Error(`Failed to fetch banners: ${response.statusText}`);
      }
      return await response.json();
    } catch (error) {
      console.error('Error fetching banners:', error);
      throw error;
    }
  },
  async fetchProducts(requestBody: ProductSearchRequest):
Promise<ProductSearchResponse> {
    try {
      const response = await fetch(`#${BASE_URL}/rest/product/search`, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify(requestBody),
      });

      if (!response.ok) {
        throw new Error(`Failed to fetch products: ${response.statusText}`);
      }

      return await response.json();
    } catch (error) {
      console.error('Error fetching products:', error);
      throw error;
    }
  }
};
```

```
    }
};
```

types/api.ts

```
// API response types
export interface Banner {
  id: string;
  message: string;
  style: 'primary' | 'warning' | 'success';
}

export interface ProductSearchResponse {
  data: { [key: string]: any }[];
  isLoading: boolean;
}

export interface ProductSearchRequest {
  includeProductOriginData: boolean;
  includeSystemBulkUploads: boolean;
}
```

App.tsx

```
import { useState, useEffect, useCallback } from 'react';
import { Button } from '@/components/ui/button';
import { Alert, AlertDescription } from '@/components/ui/alert';
import { BannerDisplay } from '@/components/BannerDisplay';
import { ProductDisplay } from '@/components/ProductDisplay';
import { api } from '@/services/api';
import type { Banner, ProductSearchResponse } from '@/types/api';
import { RefreshCw, AlertCircle } from 'lucide-react';

function App() {
  const [banners, setBanners] = useState<Banner[]>([]);
  const [productData, setProductData] = useState<ProductSearchResponse>({
    data: [],
    isLoading: true,
  });
  const [bannersLoading, setBannersLoading] = useState(true);
  const [error, setError] = useState<string | null>(null);

  const fetchBanners = useCallback(async () => {
    try {
      setBannersLoading(true);
      setError(null);
      const fetchedBanners = await api.fetchBanners();
      setBanners(fetchedBanners);
    } catch (err) {
      setError(err.message);
    }
  }, [setBanners, setError]);
}

export default App;
```

```
        setError(`Failed to load banners: ${err instanceof Error ? err.message : 'Unknown error'}`);
    } finally {
      setBannersLoading(false);
    }
}, []);
```

```
const fetchProducts = useCallback(async () => {
  try {
    setError(null);
    setProductData(prev => ({ ...prev, isLoading: true }));
  }
  const response = await api.fetchProducts({
    includeProductOriginData: false,
    includeSystemBulkUploads: false,
  });
  setProductData(response);
} catch (err) {
  setError(`Failed to load products: ${err instanceof Error ? err.message : 'Unknown error'}`);
  setProductData(prev => ({ ...prev, isLoading: false }));
}
}, []);
```

```
const refreshAll = useCallback(async () => {
  await Promise.all([fetchBanners(), fetchProducts()]);
}, [fetchBanners, fetchProducts]);
```

```
useEffect(() => {
  refreshAll();
}, [refreshAll]);
```

```
return (
  <div className="min-h-screen bg-gray-50">
    <div className="container mx-auto px-4 py-8">
      {/* Header */}
      <div className="flex items-center justify-between mb-8">
        <div>
          <h1 className="text-4xl font-bold text-gray-900">Product
Dashboard</h1>
          <p className="text-gray-600 mt-2">
            View system banners and product information
          </p>
        </div>
        <Button
          onClick={refreshAll}
          disabled={bannersLoading || productData.isLoading}
          className="flex items-center gap-2"
        >
          <RefreshCw className={`h-4 w-4 ${!(bannersLoading || productData.isLoading) ? 'animate-spin' : ''}`}>/>
          Refresh
        </Button>
      </div>
    </div>
  </div>
)
```

```
</div>

/* Error Alert */
{error && (
  <Alert variant="destructive" className="mb-6">
    <AlertCircle className="h-4 w-4" />
    <AlertDescription>{error}</AlertDescription>
  </Alert>
) {}

/* Banners Section */
{bannersLoading ? (
  <div className="mb-6">
    <div className="animate-pulse">
      <div className="h-6 bg-gray-200 rounded w-1/4 mb-4" />
      <div className="h-16 bg-gray-200 rounded" />
    </div>
  </div>
) : (
  <BannerDisplay banners={banners} />
) {}

/* Products Section */
<ProductDisplay productData={productData} />
</div>
</div>
);

}

export default App;
```

Cursor

components/Banner.tsx

```
// 12:15:app/src/components/Banner.tsx
import React from 'react';

type BannerProps = {
  id: string;
  message: string;
  style: 'primary' | 'warning' | 'success';
};

const Banner: React.FC<BannerProps> = ({ message, style }) => {
  return (
    <div className={`banner banner-${style}`}>
      {message}
    </div>
  );
};
```

```
export default Banner;
```

components/ProductList.tsx

```
// 12:15:app/src/components/ProductList.tsx
import React, { useEffect, useState } from 'react';
import axios from 'axios';

const ProductList: React.FC = () => {
  const [products, setProducts] = useState<any>([]);
  const [isLoading, setIsLoading] = useState(true);

  useEffect(() => {
    const fetchProducts = async () => {
      try {
        const response = await axios.post('http://localhost:8080/rest/product/search', {
          includeProductOriginData: false,
          includeSystemBulkUploads: false,
        });
        setProducts(response.data.data);
      } catch (error) {
        console.error('Error fetching products:', error);
      } finally {
        setIsLoading(false);
      }
    };
    fetchProducts();
  }, []);

  if (isLoading) {
    return <div>Loading products...</div>;
  }

  return (
    <div>
      <h2>Product List</h2>
      <ul>
        {products.map((product, index) => (
          <li key={index}>{JSON.stringify(product)}</li>
        ))}
      </ul>
    </div>
  );
};

export default ProductList;
```

App.tsx

```
// 12:15:app/src/App.tsx
import React, { useEffect, useState } from 'react';
import axios from 'axios';
import Banner from './components/Banner';
import ProductList from './components/ProductList';

const App: React.FC = () => {
  const [banners, setBanners] = useState<any>([]);
  const [isLoadingBanners, setIsLoadingBanners] = useState(true);

  useEffect(() => {
    const fetchBanners = async () => {
      try {
        const response = await axios.get('http://localhost:8080/rest/banners');
        setBanners(response.data);
      } catch (error) {
        console.error('Error fetching banners:', error);
      } finally {
        setIsLoadingBanners(false);
      }
    };

    fetchBanners();
  }, []);

  return (
    <div>
      <h1>My Webpage</h1>
      {isLoadingBanners ? (
        <div>Loading banners...</div>
      ) : (
        <div>
          {banners.map(banner => (
            <Banner key={banner.id} {...banner} />
          )));
        </div>
      )}
      <ProductList />
    </div>
  );
};

export default App;
```

App.css

```
/* 12:15:app/src/App.css */
.banner {
```

```
padding: 10px;
margin: 10px 0;
border-radius: 5px;
}

.banner-primary {
background-color: #007bff;
color: white;
}

.banner-warning {
background-color: #ffc107;
color: black;
}

.banner-success {
background-color: #28a745;
color: white;
}
```

Claude Opus 4

ProductDisplay.tsx

```
import React, { useState, useEffect } from 'react';

// Types
type Product = {
  [key: string]: any;
};

type ProductResponse = {
  data: Product[];
  isLoading: boolean;
};

type Banner = {
  id: string;
  message: string;
  style: 'primary' | 'warning' | 'success';
};

// API configuration
const BASE_URL = 'http://localhost:8080';

// Main Component
const ProductBannerPage = () => {
  const [products, setProducts] = useState<Product[]>([]);
  const [banners, setBanners] = useState<Banner[]>([]);
  const [isLoadingProducts, setIsLoadingProducts] = useState(true);
```

```
const [isLoadingBanners, setIsLoadingBanners] = useState(true);
const [productError, setProductError] = useState<string | null>(null);
const [bannerError, setBannerError] = useState<string | null>(null);

// Fetch products
useEffect(() => {
  const fetchProducts = async () => {
    try {
      setIsLoadingProducts(true);
      setProductError(null);

      const response = await fetch(` ${BASE_URL}/rest/product/search`, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({
          includeProductOriginData: false,
          includeSystemBulkUploads: false,
        }),
      });

      if (!response.ok) {
        throw new Error(`HTTP error! status: ${response.status}`);
      }

      const result: ProductResponse = await response.json();
      setProducts(result.data || []);
    } catch (error) {
      console.error('Error fetching products:', error);
      setProductError(error instanceof Error ? error.message : 'Failed to fetch products');
    } finally {
      setIsLoadingProducts(false);
    }
  };

  fetchProducts();
}, []);

// Fetch banners
useEffect(() => {
  const fetchBanners = async () => {
    try {
      setIsLoadingBanners(true);
      setBannerError(null);

      const response = await fetch(` ${BASE_URL}/rest/banners`, {
        method: 'GET',
        headers: {
          'Content-Type': 'application/json',
        },
      });
    }
  };
});
```

```
if (!response.ok) {
    throw new Error(`HTTP error! status: ${response.status}`);
}

const result = await response.json();
// Handle both single banner and array of banners
setBanners(Array.isArray(result) ? result : [result]);
} catch (error) {
    console.error('Error fetching banners:', error);
    setBannerError(error instanceof Error ? error.message : 'Failed to fetch
banners');
} finally {
    setIsLoadingBanners(false);
}
};

fetchBanners();
}, []);

// Banner component
const Banner = ({ banner }: { banner: Banner }) => {
    const styles = {
        primary: {
            backgroundColor: '#e3f2fd',
            borderColor: '#2196f3',
            color: '#1565c0',
        },
        warning: {
            backgroundColor: '#fff3cd',
            borderColor: '#ffc107',
            color: '#856404',
        },
        success: {
            backgroundColor: '#d4edda',
            borderColor: '#28a745',
            color: '#155724',
        },
    };
    const bannerStyle = styles[banner.style] || styles.primary;

    return (
        <div
            style={{
                padding: '16px',
                marginBottom: '16px',
                borderRadius: '8px',
                border: `1px solid ${bannerStyle.borderColor}`,
                backgroundColor: bannerStyle.backgroundColor,
                color: bannerStyle.color,
                display: 'flex',
                alignItems: 'center',
                gap: '12px',
            }}
    
```

```
>
  {banner.style === 'warning' && (
    <svg width="20" height="20" viewBox="0 0 20 20" fill="currentColor">
      <path d="M10 2C5.58 2 2 5.58 2 10s3.58 8 8 8 8-3.58 8-8-3.58-8-8zm1
13h-2v-2h2v2zm0-4h-2V7h2v4z"/>
    </svg>
  )}
  {banner.style === 'success' && (
    <svg width="20" height="20" viewBox="0 0 20 20" fill="currentColor">
      <path d="M10 2C5.58 2 2 5.58 2 10s3.58 8 8 8 8-3.58 8-8-3.58-8-8zm-1
12l-3-3 1.41-1.41L9 11.17l4.59-4.58L15 8l-6 6z"/>
    </svg>
  )}
  {banner.style === 'primary' && (
    <svg width="20" height="20" viewBox="0 0 20 20" fill="currentColor">
      <path d="M10 2C5.58 2 2 5.58 2 10s3.58 8 8 8 8-3.58 8-8-3.58-8-8zm1
13h-2v-2h2v2zm0-4h-2V7h2v4z"/>
    </svg>
  )}
  <span style={{ flex: 1 }}>{banner.message}</span>
</div>
);
};

// Product card component
const ProductCard = ({ product }: { product: Product }) => {
  // Extract common product fields if they exist
  const productId = product.id || product.productId || 'N/A';
  const productName = product.name || product.productName || product.title ||
  'Unnamed Product';
  const productDescription = product.description || product.productDescription ||
  '';
  const productPrice = product.price || product.productPrice || null;
  const productImage = product.image || product.imageUrl || product.productImage ||
  null;

  return (
    <div
      style={{
        border: '1px solid #e0e0e0',
        borderRadius: '8px',
        padding: '16px',
        backgroundColor: '#fff',
        boxShadow: '0 2px 4px rgba(0,0,0,0.1)',
        transition: 'transform 0.2s, box-shadow 0.2s',
        cursor: 'pointer',
      }}
      onMouseEnter={(e) => {
        e.currentTarget.style.transform = 'translateY(-2px)';
        e.currentTarget.style.boxShadow = '0 4px 8px rgba(0,0,0,0.15)';
      }}
      onMouseLeave={(e) => {
        e.currentTarget.style.transform = 'translateY(0)';
        e.currentTarget.style.boxShadow = '0 2px 4px rgba(0,0,0,0.1)';
      }}
    >
      {product.name}
    </div>
  );
};
```

```
        })}
      >
    {productImage && (
      <div
        style={{
          width: '100%',
          height: '200px',
          backgroundColor: '#f5f5f5',
          borderRadius: '4px',
          marginBottom: '12px',
          display: 'flex',
          alignItems: 'center',
          justifyContent: 'center',
          overflow: 'hidden',
        }}
      >
        <img
          src={productImage}
          alt={productName}
          style={{
            maxWidth: '100%',
            maxHeight: '100%',
            objectFit: 'cover',
          }}
        >
        onError={(e) => {
          e.currentTarget.style.display = 'none';
          e.currentTarget.parentElement!.innerHTML = '<div style="color:#999;">No image</div>';
        }}
      />
    </div>
  )}

  <h3 style={{ margin: '0 0 8px 0', fontSize: '18px', color: '#333' }}>
    {productName}
  </h3>

  {productDescription && (
    <p style={{ margin: '0 0 12px 0', color: '#666', fontSize: '14px', lineHeight: '1.5' }}>
      {productDescription}
    </p>
  )}

  {productPrice !== null && (
    <div style={{ fontSize: '20px', fontWeight: 'bold', color: '#2196f3' }}>
      ${typeof productPrice === 'number' ? productPrice.toFixed(2) : productPrice}
    </div>
  )}

  <div style={{ marginTop: '12px', fontSize: '12px', color: '#999' }}>
    ID: {productId}
  </div>
```

```
        </div>
    );
};

// Loading spinner component
const LoadingSpinner = () => (
    <div style={{ display: 'flex', justifyContent: 'center', padding: '20px' }}>
        <div
            style={{
                width: '40px',
                height: '40px',
                border: '4px solid #f3f3f3',
                borderTop: '4px solid #2196f3',
                borderRadius: '50%',
                animation: 'spin 1s linear infinite',
            }}
        />
    </div>
);

// Error component
const ErrorMessage = ({ error }: { error: string }) => (
    <div
        style={{
            padding: '16px',
            backgroundColor: '#ffebee',
            border: '1px solid #f44336',
            borderRadius: '4px',
            color: '#c62828',
            marginBottom: '16px',
        }}
    >
        <strong>Error:</strong> {error}
    </div>
);

return (
    <div style={{ minHeight: '100vh', backgroundColor: '#f5f5f5' }}>
        {/* Add spinner animation */}
        <style>{
            @keyframes spin {
                0% { transform: rotate(0deg); }
                100% { transform: rotate(360deg); }
            }
        }</style>

        {/* Header */}
        <header
            style={{
                backgroundColor: '#2196f3',
                color: 'white',
                padding: '20px',
                boxShadow: '0 2px 4px rgba(0,0,0,0.1)',
            }}
        >
```

```
>
<div style={{ maxWidth: '1200px', margin: '0 auto' }}>
  <h1 style={{ margin: 0, fontSize: '28px' }}>Product Catalog</h1>
</div>
</header>

{/* Main Content */}
<main style={{ maxWidth: '1200px', margin: '0 auto', padding: '20px' }}>
  {/* Banners Section */}
  <section style={{ marginBottom: '40px' }}>
    {isLoadingBanners ? (
      <LoadingSpinner />
    ) : bannerError ? (
      <ErrorMessage error={bannerError} />
    ) : (
      banners.map((banner) => (
        <Banner key={banner.id} banner={banner} />
      ))
    )}
  </section>

  {/* Products Section */}
  <section>
    <h2 style={{ marginBottom: '24px', color: '#333' }}>Products</h2>

    {isLoadingProducts ? (
      <LoadingSpinner />
    ) : productError ? (
      <ErrorMessage error={productError} />
    ) : products.length === 0 ? (
      <div
        style={{
          textAlign: 'center',
          padding: '40px',
          backgroundColor: '#fff',
          borderRadius: '8px',
          color: '#666',
        }}
      >
        No products found.
      </div>
    ) : (
      <div
        style={{
          display: 'grid',
          gridTemplateColumns: 'repeat(auto-fill, minmax(280px, 1fr))',
          gap: '24px',
        }}
      >
        {products.map((product, index) => (
          <ProductCard key={product.id || index} product={product} />
        ))}
      </div>
    )}
  </section>
</main>
```

```
</section>
</main>

{/* Footer */}
<footer
  style={{
    backgroundColor: '#333',
    color: '#fff',
    padding: '20px',
    marginTop: '60px',
    textAlign: 'center',
  }}
>
  <div style={{ maxWidth: '1200px', margin: '0 auto' }}>
    <p style={{ margin: 0 }}>© 2024 Product Catalog. All rights reserved.</p>
  </div>
</footer>
</div>
);
};
```

Claude Sonnet 4

ProductDashboard.tsx

```
import React, { useState, useEffect } from 'react';

// Types
type Banner = {
  id: string;
  message: string;
  style: 'primary' | 'warning' | 'success';
};

type ProductSearchResponse = {
  data: { [key: string]: any }[];
  isLoading: boolean;
};

type ApiState<T> = {
  data: T | null;
  loading: boolean;
  error: string | null;
};

// Custom hook for API calls
const useApi = <T, >(url: string, options?: RequestInit): ApiState<T> => {
  const [state, setState] = useState<ApiState<T>>({
    data: null,
```

```
loading: true,
error: null
});

useEffect(() => {
  const fetchData = async () => {
    try {
      setState(prev => ({ ...prev, loading: true, error: null }));
    }

    const response = await fetch(`http://localhost:8080${url}`, {
      ...options,
      headers: {
        'Content-Type': 'application/json',
        ...options?.headers,
      },
    });

    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`);
    }

    const result = await response.json();
    setState({ data: result, loading: false, error: null });
  } catch (error) {
    setState({
      data: null,
      loading: false,
      error: error instanceof Error ? error.message : 'An error occurred'
    });
  }
};

fetchData();
}, [url, JSON.stringify(options)]);

return state;
};

// Banner Component
const BannerComponent: React.FC<{ banner: Banner; onDismiss: (id: string) => void }> = ({ banner, onDismiss }) => {
  const getBannerStyles = (style: Banner['style']) => {
    const baseStyles: React.CSSProperties = {
      padding: '12px 16px',
      borderRadius: '8px',
      marginBottom: '12px',
      display: 'flex',
      justifyContent: 'space-between',
      alignItems: 'center',
      fontSize: '14px',
      fontWeight: '500',
    };
    if (style === 'info') {
      baseStyles.color = '#007bff';
      baseStyles.backgroundColor = '#d9e1f2';
    } else if (style === 'success') {
      baseStyles.color = '#28a745';
      baseStyles.backgroundColor = '#d4edbc';
    } else if (style === 'warning') {
      baseStyles.color = '#ffc107';
      baseStyles.backgroundColor = '#fff3cd';
    } else if (style === 'danger') {
      baseStyles.color = '#dc3545';
      baseStyles.backgroundColor = '#ffcba1';
    }
    return baseStyles;
  };
  return (
    <div style={getBannerStyles(banner.style)}>
      {banner.message}
      <button onClick={() => onDismiss(banner.id)}>Close</button>
    </div>
  );
};
```

```
        boxShadow: '0 2px 4px rgba(0, 0, 0, 0.1)',  
    };  
  
    switch (style) {  
        case 'primary':  
            return {  
                ...baseStyles,  
                backgroundColor: '#3b82f6',  
                color: '#ffffff',  
                border: '1px solid #2563eb',  
            };  
        case 'warning':  
            return {  
                ...baseStyles,  
                backgroundColor: '#fbbf24',  
                color: '#92400e',  
                border: '1px solid #f59e0b',  
            };  
        case 'success':  
            return {  
                ...baseStyles,  
                backgroundColor: '#10b981',  
                color: '#ffffff',  
                border: '1px solid #059669',  
            };  
        default:  
            return baseStyles;  
    }  
};  
  
return (  
    <div style={getBannerStyles(banner.style)}>  
        <span>{banner.message}</span>  
        <button  
            onClick={() => onDismiss(banner.id)}  
            style={{  
                background: 'none',  
                border: 'none',  
                color: 'inherit',  
                cursor: 'pointer',  
                fontSize: '16px',  
                padding: '0 4px',  
                opacity: 0.8,  
                transition: 'opacity 0.2s ease',  
            }}  
            onMouseEnter={(e) => (e.currentTarget.style.opacity = '1')}            onMouseLeave={(e) => (e.currentTarget.style.opacity = '0.8')}>  
            >  
            ×  
        </button>  
    </div>  
);  
};
```

```
// Product Card Component
const ProductCard: React.FC<{ product: { [key: string]: any } }> = ({ product }) => {
  const cardStyles: React.CSSProperties = {
    backgroundColor: '#ffffff',
    border: '1px solid #e5e7eb',
    borderRadius: '8px',
    padding: '16px',
    boxShadow: '0 1px 3px rgba(0, 0, 0, 0.1)',
    transition: 'transform 0.2s ease, box-shadow 0.2s ease',
    cursor: 'pointer',
  };
  const handleMouseEnter = (e: React.MouseEvent<HTMLDivElement>) => {
    e.currentTarget.style.transform = 'translateY(-2px)';
    e.currentTarget.style.boxShadow = '0 4px 12px rgba(0, 0, 0, 0.15)';
  };
  const handleMouseLeave = (e: React.MouseEvent<HTMLDivElement>) => {
    e.currentTarget.style.transform = 'translateY(0)';
    e.currentTarget.style.boxShadow = '0 1px 3px rgba(0, 0, 0, 0.1)';
  };
  // Extract common product fields
  const productName = product.name || product.title || product.productName || 'Unnamed Product';
  const productId = product.id || product.productId || 'N/A';
  const price = product.price || product.cost || product.amount;
  const description = product.description || product.summary || '';
  return (
    <div
      style={cardStyles}
      onMouseEnter={handleMouseEnter}
      onMouseLeave={handleMouseLeave}>
      <div style={{ marginBottom: '12px' }}>
        <h3 style={{
          margin: '0 0 8px 0',
          fontSize: '18px',
          fontWeight: '600',
          color: '#111827',
          lineHeight: '1.4',
        }}>
          {productName}
        </h3>
        <p style={{
          margin: '0',
          fontSize: '12px',
          color: '#6b7280',
          fontFamily: 'monospace',
        }}>
          ID: {productId}
        </p>
      </div>
    </div>
  );
}
```

```
</div>

{price && (
  <div style={{ 
    fontSize: '16px',
    fontWeight: '600',
    color: '#059669',
    marginBottom: '12px',
  }}>
    ${typeof price === 'number' ? price.toFixed(2) : price}
  </div>
)}

{description && (
  <p style={{ 
    margin: '0 0 16px 0',
    fontSize: '14px',
    color: '#4b5563',
    lineHeight: '1.5',
  }}>
    {description.length > 100 ? `${description.substring(0, 100)}...` :
  description}
  </p>
)

/* Display other product fields */
<div style={{ fontSize: '12px', color: '#6b7280' }}>
  {Object.entries(product)
    .filter(([key]) => !['name', 'title', 'productName', 'id', 'productId',
  'price', 'cost', 'amount', 'description', 'summary'].includes(key))
    .slice(0, 3)
    .map(([key, value]) => (
      <div key={key} style={{ marginBottom: '4px' }}>
        <strong>{key}</strong> {String(value).substring(0, 50)}
      {String(value).length > 50 ? '...' : ''}
      </div>
    )))
  </div>
</div>
);

// Loading Component
const LoadingSpinner: React.FC = () => (
  <div style={{ 
    display: 'flex',
    justifyContent: 'center',
    alignItems: 'center',
    padding: '40px',
  }}>
    <div style={{ 
      width: '40px',
      height: '40px',
      border: '4px solid #f3f4f6',
    }}>

```

```
borderTop: '4px solid #3b82f6',
borderRadius: '50%',
animation: 'spin 1s linear infinite',
}} />
<style>{
  @keyframes spin {
    0% { transform: rotate(0deg); }
    100% { transform: rotate(360deg); }
  }
`}</style>
</div>
);

// Error Component
const ErrorMessage: React.FC<{ message: string; onRetry?: () => void }> = ({ message, onRetry }) => (
  <div style={{
    backgroundColor: '#fef2f2',
    border: '1px solid #fecaca',
    borderRadius: '8px',
    padding: '16px',
    margin: '16px 0',
    color: '#dc2626',
  }}>
    <div style={{ display: 'flex', alignItems: 'center', marginBottom: '8px' }}>
      <span style={{ fontSize: '18px', marginRight: '8px' }}>⚠</span>
      <strong>Error</strong>
    </div>
    <p style={{ margin: '0 0 12px 0' }}>{message}</p>
    {onRetry && (
      <button
        onClick={onRetry}
        style={{
          backgroundColor: '#dc2626',
          color: 'white',
          border: 'none',
          padding: '8px 16px',
          borderRadius: '4px',
          cursor: 'pointer',
          fontSize: '14px',
        }}
      >
        Retry
      </button>
    )}
  </div>
);

// Main Dashboard Component
const ProductDashboard: React.FC = () => {
  const [dismissedBanners, setDismissedBanners] = useState<Set<string>>(new Set());
}

// Fetch banners
```

```
const bannersState = useApi<Banner[]>('/rest/banners', {
  method: 'GET',
});

// Fetch products
const productsState = useApi<ProductSearchResponse>('/rest/product/search', {
  method: 'POST',
  body: JSON.stringify({
    includeProductOriginData: false,
    includeSystemBulkUploads: false,
  }),
});

const handleDismissBanner = (bannerId: string) => {
  setDismissedBanners(prev => new Set([...prev, bannerId]));
};

const visibleBanners = bannersState.data?.filter(banner =>
!dismissedBanners.has(banner.id)) || [];

const containerStyles: React.CSSProperties = {
  maxWidth: '1200px',
  margin: '0 auto',
  padding: '20px',
  fontFamily: 'system-ui, -apple-system, sans-serif',
  backgroundColor: '#f9fafb',
  minHeight: '100vh',
};

const headerStyles: React.CSSProperties = {
  textAlign: 'center',
  marginBottom: '32px',
  color: '#111827',
};

const sectionStyles: React.CSSProperties = {
  marginBottom: '32px',
};

const gridStyles: React.CSSProperties = {
  display: 'grid',
  gridTemplateColumns: 'repeat(auto-fill, minmax(300px, 1fr))',
  gap: '20px',
  marginTop: '20px',
};

return (
  <div style={containerStyles}>
    <header style={headerStyles}>
      <h1 style={{ fontSize: '32px', fontWeight: '700', margin: '0 0 8px 0' }>
        Product Dashboard
      </h1>
      <p style={{ fontSize: '16px', color: '#6b7280', margin: '0' }>
        Browse our product catalog and important announcements
      </p>
    </header>
  </div>
)
```

```
</p>
</header>

{/* Banners Section */}
<section style={sectionStyles}>
  {bannersState.loading && <LoadingSpinner />}
  {bannersState.error && (
    <ErrorMessage
      message={`Failed to load banners: ${bannersState.error}`}
    />
  )}
  {visibleBanners.length > 0 && (
    <div>
      {visibleBanners.map(banner => (
        <BannerComponent
          key={banner.id}
          banner={banner}
          onDismiss={handleDismissBanner}
        />
      )))
    </div>
  )}
</section>

{/* Products Section */}
<section style={sectionStyles}>
  <h2 style={{
    fontSize: '24px',
    fontWeight: '600',
    margin: '0 0 16px 0',
    color: '#111827'
  }}>
    Products
  </h2>

  {productsState.loading && <LoadingSpinner />}

  {productsState.error && (
    <ErrorMessage
      message={`Failed to load products: ${productsState.error}`}
    />
  )}

  {productsState.data && (
    <>
      {productsState.data.isLoading && (
        <div style={{
          backgroundColor: '#fffbeb',
          border: '1px solid #fbbf24',
          borderRadius: '8px',
          padding: '12px 16px',
          marginBottom: '16px',
          color: '#92400e',
          fontSize: '14px',
        }}>
          {productsState.data.items.map(item => (
            <ProductCard
              key={item.id}
              item={item}
            />
          ))}
        </div>
      )}
  )}
```

```
        })>
        ⚡ Products are still loading on the server...
    </div>
    )}

    <div style={{>
        display: 'flex',
        justifyContent: 'space-between',
        alignItems: 'center',
        marginBottom: '16px',
        padding: '12px 16px',
        backgroundColor: '#ffffff',
        borderRadius: '8px',
        border: '1px solid #e5e7eb',
    }}>
        <span style={{ fontSize: '14px', color: '#6b7280' }}>
            Showing {productsState.data.data.length} products
        </span>
        <span style={{ fontSize: '12px', color: '#9ca3af' }}>
            Last updated: {new Date().toLocaleTimeString()}
        </span>
    </div>

{productsState.data.data.length === 0 ? (
    <div style={{>
        textAlign: 'center',
        padding: '40px',
        backgroundColor: '#ffffff',
        borderRadius: '8px',
        border: '1px solid #e5e7eb',
    }}>
        <div style={{ fontSize: '48px', marginBottom: '16px' }}>∅</div>
        <h3 style={{ color: '#6b7280', margin: '0 0 8px 0' }}>No products
    found</h3>
        <p style={{ color: '#9ca3af', margin: '0' }}>
            Try adjusting your search criteria or check back later.
        </p>
    </div>
) : (
    <div style={gridStyles}>
        {productsState.data.data.map((product, index) => (
            <ProductCard
                key={product.id || product.productId || index}
                product={product}
            />
        )))
    </div>
)
    </>
)
    </section>
</div>
);
};
```

```
export default ProductDashboard;
```

Task 5

GPT 4.1

EditLocationPage.tsx

```
import React, { useState } from "react";

// Dummy country list for demo purposes
const countries = [
  "UNITED STATES",
  "CANADA",
  "MEXICO",
];

const tempZones = [
  "All",
  "Ambient",
  "Chilled",
  "Frozen",
];

export default function EditLocationPage() {
  // State for form fields
  const [general, setGeneral] = useState({
    company: "JR Simplot Company",
    name: "JR Simplot Company - USA - Moses Lake (WHS)",
    abbreviation: "",
    number: "25848",
    externalNumber: "",
    globalLocationNumber: "",
    tmsIntegrationLocationNumber: "",
    tmsIntegrationActive: true,
    tmsIntegrationOptions: {
      finalMile: false,
      localLoad: false,
      redCap: false,
      pilotUpload: false,
      supportBackorders: false,
      defaultUOM: false,
      ignoreUOM: false,
      loadBuildASAP: false,
      lotNumberRequired: false,
    },
    locationType: "Shipping",
    usdaFadNumber: "",
  });
}
```

```
const [facility, setFacility] = useState({
  tempZone: "All",
  link: "",
  facility: "8963631 - JR Simplot Company - USA-Moses Lake (WHS)/USA-Moses Lake
(WHS) - 2107 Road O Northeast Moses Lake WA 98837",
});

const [address, setAddress] = useState({
  line1: "2107 Road O Northeast",
  line2: "",
  city: "Moses Lake",
  state: "WA",
  postalCode: "98837",
  country: "UNITED STATES",
});

const [contacts, setContacts] = useState([
  // Example: { role: "", firstName: "", lastName: "", email: "", phone: "", fax: "" }
]);

const [orderQuantities, setOrderQuantities] = useState([
  // Example: { tempZone: "All", maxOQ: "", oqUOM: "", effectiveDate: "" }
]);

const [freight, setFreight] = useState({
  details: "",
  washoutRequired: false,
  shippingPlatform: "",
  requirements: "",
});

const [comments, setComments] = useState({
  orderComment: "",
  newComment: "",
});

// Handlers
function handleGeneralChange(e: React.ChangeEvent<HTMLInputElement | HTMLSelectElement>) {
  const { name, value, type, checked } = e.target;
  if (name in general.tmsIntegrationOptions) {
    setGeneral((prev) => ({
      ...prev,
      tmsIntegrationOptions: {
        ...prev.tmsIntegrationOptions,
        [name]: checked,
      },
    }));
  } else if (type === "checkbox") {
    setGeneral((prev) => ({ ...prev, [name]: checked }));
  } else {
    setGeneral((prev) => ({ ...prev, [name]: value }));
  }
}
```

```
}

function handleAddressChange(e: React.ChangeEvent<HTMLInputElement | HTMLSelectElement>) {
  const { name, value } = e.target;
  setAddress((prev) => ({ ...prev, [name]: value }));
}

function handleFacilityChange(e: React.ChangeEvent<HTMLSelectElement>) {
  const { name, value } = e.target;
  setFacility((prev) => ({ ...prev, [name]: value }));
}

function handleFreightChange(e: React.ChangeEvent<HTMLInputElement | HTMLSelectElement | HTMLTextAreaElement>) {
  const { name, value, type, checked } = e.target;
  setFreight((prev) => ({
    ...prev,
    [name]: type === "checkbox" ? checked : value,
  }));
}

function handleCommentsChange(e: React.ChangeEvent<HTMLInputElement | HTMLTextAreaElement>) {
  const { name, value } = e.target;
  setComments((prev) => ({ ...prev, [name]: value }));
}

// ...handlers for contacts and order quantities would go here...

// Submit handler, not implemented
function handleSubmit(e: React.FormEvent) {
  e.preventDefault();
  // Submit form logic here
  alert("Save functionality not implemented in this demo.");
}

// Layout
return (
  <form className="edit-location-page" style={{ fontFamily: "Arial, sans-serif",
  fontSize: 13, background: "#f9f9f6" }} onSubmit={handleSubmit}>
    {/* Header */}
    <div style={{ background: "#b6d47d", padding: "8px 20px", fontWeight: 600,
    color: "#234", fontSize: 14 }}>
      JR Simplot Company - USA - Moses Lake (WHS)/USA-Moses Lake (WHS) (25848)
      <span style={{ float: "right", fontWeight: 400 }}>
        <button type="button" style={{ marginRight: 7 }}>Download</button>
        <button type="submit">Save</button>
      </span>
    </div>
    {/* General */}
    <section style={{ borderBottom: "1px solid #e2e2e2", padding: "10px 20px" }}>
```

```
<div style={{ fontWeight: 600, marginBottom: 8 }}>General</div>
<div style={{ display: "flex", gap: 20, alignItems: "center", flexWrap: "wrap" }}>
    <label>
        Company:
        <input style={{ width: 240 }} name="company" value={general.company} onChange={handleGeneralChange} />
    </label>
    <label>
        Name:
        <input style={{ width: 320 }} name="name" value={general.name} onChange={handleGeneralChange} />
    </label>
    <label>
        Abbreviation:
        <input style={{ width: 120 }} name="abbreviation" value={general.abbreviation} onChange={handleGeneralChange} />
    </label>
    <label>
        Number:
        <input style={{ width: 90 }} name="number" value={general.number} onChange={handleGeneralChange} />
    </label>
    <label>
        External Number:
        <input style={{ width: 90 }} name="externalNumber" value={general.externalNumber} onChange={handleGeneralChange} />
    </label>
    <label>
        GLN:
        <input style={{ width: 140 }} name="globalLocationNumber" value={general.globalLocationNumber} onChange={handleGeneralChange} />
    </label>
    <label>
        TMS Location Number:
        <input style={{ width: 140 }} name="tmsIntegrationLocationNumber" value={general.tmsIntegrationLocationNumber} onChange={handleGeneralChange} />
    </label>
</div>
<div style={{ marginTop: 8, display: "flex", gap: 18, alignItems: "center", flexWrap: "wrap" }}>
    <label style={{ color: "#234" }}>
        TMS Integration
        <input type="checkbox" name="tmsIntegrationActive" checked={general.tmsIntegrationActive} onChange={handleGeneralChange} style={{ marginLeft: 6 }} />
    </label>
    {Object.keys(general.tmsIntegrationOptions).map((k) => (
        <label key={k} style={{ fontWeight: 400 }}>
            <input type="checkbox" name={k} checked={general.tmsIntegrationOptions[k]} onChange={handleGeneralChange} />
            {" " + k.replace(/([A-Z])/g, " $1")}
        </label>
    )))
}
```

```
</div>
<div style={{ marginTop: 12 }}>
  <label>
    Location Type:
    <select name="locationType" value={general.locationType} onChange={handleGeneralChange} style={{ marginLeft: 5 }}>
      <option>Shipping</option>
      <option>Warehouse</option>
      <option>Receiving</option>
    </select>
  </label>
  <label style={{ marginLeft: 18 }}>
    USDA FAD Number:
    <input style={{ width: 120 }} name="usdaFadNumber" value={general.usdaFadNumber} onChange={handleGeneralChange} />
  </label>
</div>
</section>

/* Facility */
<section style={{ borderBottom: "1px solid #e2e2e2", padding: "10px 20px" }}>
  <div style={{ fontWeight: 600, marginBottom: 8 }}>Facility</div>
  <label>
    Temperature Zone:
    <select name="tempZone" value={facility.tempZone} onChange={handleFacilityChange} style={{ marginLeft: 5 }}>
      {tempZones.map((z) => <option key={z}>{z}</option>)}
    </select>
  </label>
  <label style={{ marginLeft: 25 }}>
    Link:
    <input style={{ width: 180 }} name="link" value={facility.link} onChange={handleFacilityChange} />
  </label>
  <div style={{ marginTop: 8, color: "#234" }}>
    Facility: {facility.facility}
  </div>
</section>

/* Address */
<section style={{ borderBottom: "1px solid #e2e2e2", padding: "10px 20px" }}>
  <div style={{ fontWeight: 600, marginBottom: 8 }}>Address</div>
  <div style={{ display: "flex", gap: 20, alignItems: "center", flexWrap: "wrap" }}>
    <label>
      Line 1:
      <input style={{ width: 320 }} name="line1" value={address.line1} onChange={handleAddressChange} />
    </label>
    <label>
      Line 2:
      <input style={{ width: 320 }} name="line2" value={address.line2} />
    </label>
  </div>
</section>
```

```
onChange={handleAddressChange} />
    </label>
    <label>
        City:
        <input style={{ width: 140 }} name="city" value={address.city}
onChange={handleAddressChange} />
    </label>
    <label>
        State:
        <input style={{ width: 50 }} name="state" value={address.state}
onChange={handleAddressChange} />
    </label>
    <label>
        Postal Code:
        <input style={{ width: 90 }} name="postalCode" value=
{address.postalCode} onChange={handleAddressChange} />
    </label>
    <label>
        Country:
        <select name="country" value={address.country} onChange=
{handleAddressChange}>
            {countries.map((c) => <option key={c}>{c}</option>)}
        </select>
    </label>
</div>
<div style={{ marginTop: 12 }}>
    {/* Fake embedded map */}
    <iframe
        title="Location Map"
        width="600"
        height="130"
        style={{ border: "1px solid #d2d2d2", borderRadius: 8 }}
        src="https://maps.google.com/maps?
q=Moses%20Lake%20WA%2098837&t=&z=8&ie=UTF8&iwloc=&output=embed"
    />
</div>
</section>

/* Contacts */
<section style={{ borderBottom: "1px solid #e2e2e2", padding: "10px 20px" }}>
    <div style={{ fontWeight: 600, marginBottom: 8 }}>Contacts</div>
    <table style={{ width: "100%", borderCollapse: "collapse", background:
"#fafafad" }}>
        <thead>
            <tr>
                <th>Role</th><th>First Name</th><th>Last Name</th><th>Email</th>
        <th>Phone</th><th>Fax</th><th>Actions</th>
            </tr>
        </thead>
        <tbody>
            {contacts.map((c, i) => (
                <tr key={i}>
                    <td><input value={c.role} /></td>
```

```
        <td><input value={c.firstName} /></td>
        <td><input value={c.lastName} /></td>
        <td><input value={c.email} /></td>
        <td><input value={c.phone} /></td>
        <td><input value={c.fax} /></td>
        <td>
            <button type="button">Edit</button>
            <button type="button">Delete</button>
        </td>
    </tr>
))
)
</tbody>
</table>
<button type="button" style={{ marginTop: 6 }}>Add Contact</button>
</section>

/* Order Quantities */
<section style={{ borderBottom: "1px solid #e2e2e2", padding: "10px 20px" }}>
    <div style={{ fontWeight: 600, marginBottom: 8 }}>Order Quantities</div>
    <table style={{ width: "100%", borderCollapse: "collapse", background: "#fafafad" }}>
        <thead>
            <tr>
                <th>Temperature Zone</th>
                <th>Max OQ</th>
                <th>OQ UOM</th>
                <th>Effective Date</th>
                <th>Actions</th>
            </tr>
        </thead>
        <tbody>
            {orderQuantities.map((oq, i) => (
                <tr key={i}>
                    <td><input value={oq.tempZone} /></td>
                    <td><input value={oq.maxOQ} /></td>
                    <td><input value={oq.oqUOM} /></td>
                    <td><input value={oq.effectiveDate} type="date" /></td>
                    <td>
                        <button type="button">Edit</button>
                        <button type="button">Delete</button>
                    </td>
                </tr>
            ))
)
</tbody>
</table>
<button type="button" style={{ marginTop: 6 }}>Add OQ</button>
</section>

/* Freight */
<section style={{ borderBottom: "1px solid #e2e2e2", padding: "10px 20px" }}>
    <div style={{ fontWeight: 600, marginBottom: 8 }}>Freight</div>
    <div style={{ display: "flex", gap: 20, alignItems: "center" }}>
```

```
<label>
  Details:
  <input style={{ width: 180 }} name="details" value={freight.details}
onChange={handleFreightChange} />
</label>
<label>
  Washout Required:
  <input type="checkbox" name="washoutRequired" checked=
{freight.washoutRequired} onChange={handleFreightChange} />
</label>
<label>
  Shipping Platform:
  <input style={{ width: 120 }} name="shippingPlatform" value=
{freight.shippingPlatform} onChange={handleFreightChange} />
</label>
</div>
<div style={{ marginTop: 12 }}>
  Requirements:
  <textarea name="requirements" value={freight.requirements} onChange=
{handleFreightChange} style={{ width: "60%", height: 40 }} />
</div>
</section>

/* Comments */
<section style={{ padding: "10px 20px" }}>
  <div style={{ fontWeight: 600, marginBottom: 8 }}>Comments</div>
  <label>
    New Comment:
    <input style={{ width: "80%" }} name="newComment" value=
{comments.newComment} onChange={handleCommentsChange} />
  </label>
  <div style={{ marginTop: 8 }}>
    Order Comments:
    <textarea name="orderComment" value={comments.orderComment} onChange=
{handleCommentsChange} style={{ width: "80%", height: 40 }} />
  </div>
</section>
</form>
);
}
```

Same.dev

components/ui/accordion.tsx

```
import * as React from "react"
import * as AccordionPrimitive from "@radix-ui/react-accordion"
import { ChevronDown } from "lucide-react"

import { cn } from "@/lib/utils"
```

```
const Accordion = AccordionPrimitive.Root

const AccordionItem = React.forwardRef<
  React.ElementRef<typeof AccordionPrimitive.Item>,
  React.ComponentPropsWithoutRef<typeof AccordionPrimitive.Item>
>(({ className, ...props }, ref) => (
  <AccordionPrimitive.Item
    ref={ref}
    className={cn("border-b", className)}
    {...props}
  />
))
AccordionItem.displayName = "AccordionItem"

const AccordionTrigger = React.forwardRef<
  React.ElementRef<typeof AccordionPrimitive.Trigger>,
  React.ComponentPropsWithoutRef<typeof AccordionPrimitive.Trigger>
>(({ className, children, ...props }, ref) => (
  <AccordionPrimitive.Header className="flex">
    <AccordionPrimitive.Trigger
      ref={ref}
      className={cn(
        "flex flex-1 items-center justify-between py-4 text-sm font-medium",
        "transition-all hover:underline text-left [&[data-state=open]>svg]:rotate-180",
        className
      )}
      {...props}
    >
      {children}
      <ChevronDown className="h-4 w-4 shrink-0 text-muted-foreground transition-transform duration-200" />
    </AccordionPrimitive.Trigger>
  </AccordionPrimitive.Header>
))
AccordionTrigger.displayName = AccordionPrimitive.Trigger.displayName

const AccordionContent = React.forwardRef<
  React.ElementRef<typeof AccordionPrimitive.Content>,
  React.ComponentPropsWithoutRef<typeof AccordionPrimitive.Content>
>(({ className, children, ...props }, ref) => (
  <AccordionPrimitive.Content
    ref={ref}
    className="overflow-hidden text-sm data-[state=closed]:animate-accordion-up data-[state=open]:animate-accordion-down"
    {...props}
  >
    <div className={cn("pb-4 pt-0", className)}>{children}</div>
  </AccordionPrimitive.Content>
))
AccordionContent.displayName = AccordionPrimitive.Content.displayName

export { Accordion, AccordionItem, AccordionTrigger, AccordionContent }
```

components/ui/button.tsx

```
import * as React from "react"
import { Slot } from "@radix-ui/react-slot"
import { cva, type VariantProps } from "class-variance-authority"

import { cn } from "@/lib/utils"

const buttonVariants = cva(
  "inline-flex items-center justify-center gap-2 whitespace nowrap rounded-md
text-sm font-medium transition-colors focus-visible:outline-none focus-
visible:ring-1 focus-visible:ring-ring disabled:pointer-events-none
disabled:opacity-50 [&_svg]:pointer-events-none [&_svg]:size-4 [&_svg]:shrink-0",
  {
    variants: {
      variant: {
        default:
          "bg-primary text-primary-foreground shadow hover:bg-primary/90",
        destructive:
          "bg-destructive text-destructive-foreground shadow-sm hover:bg-
destructive/90",
        outline:
          "border border-input bg-background shadow-sm hover:bg-accent hover:text-
accent-foreground",
        secondary:
          "bg-secondary text-secondary-foreground shadow-sm hover:bg-
secondary/80",
        ghost: "hover:bg-accent hover:text-accent-foreground",
        link: "text-primary underline-offset-4 hover:underline",
      },
      size: {
        default: "h-9 px-4 py-2",
        sm: "h-8 rounded-md px-3 text-xs",
        lg: "h-10 rounded-md px-8",
        icon: "h-9 w-9",
      },
    },
    defaultVariants: {
      variant: "default",
      size: "default",
    },
  },
)

export interface ButtonProps
  extends React.ButtonHTMLAttributes<HTMLButtonElement>,
  VariantProps<typeof buttonVariants> {
  asChild?: boolean
}

const Button = React.forwardRef<HTMLButtonElement, ButtonProps>(
  ({ className, variant, size, asChild = false, ...props }, ref) => {
    const Comp = asChild ? Slot : "button"
```

```
        return (
          <Comp
            className={cn(buttonVariants({ variant, size, className }))}
            ref={ref}
            {...props}
          />
        )
      }
    )
  }
Button.displayName = "Button"

export { Button, buttonVariants }
```

components/ui/card.tsx

```
import * as React from "react"

import { cn } from "@/lib/utils"

const Card = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => (
  <div
    ref={ref}
    className={cn(
      "rounded-xl border bg-card text-card-foreground shadow",
      className
    )}
    {...props}
  />
))
Card.displayName = "Card"

const CardHeader = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => (
  <div
    ref={ref}
    className={cn("flex flex-col space-y-1.5 p-6", className)}
    {...props}
  />
))
CardHeader.displayName = "CardHeader"

const CardTitle = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => (
  <div
    ref={ref}
    {...props}
  />
))
```

```
      className={cn("font-semibold leading-none tracking-tight", className)}
      {...props}
    />
  ))
CardTitle.displayName = "CardTitle"

const CardDescription = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => (
  <div
    ref={ref}
    className={cn("text-sm text-muted-foreground", className)}
    {...props}
  />
))
CardDescription.displayName = "CardDescription"

constCardContent = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => (
  <div ref={ref} className={cn("p-6 pt-0", className)} {...props} />
))
CardContent.displayName = "CardContent"

const CardFooter = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => (
  <div
    ref={ref}
    className={cn("flex items-center p-6 pt-0", className)}
    {...props}
  />
))
CardFooter.displayName = "CardFooter"

export { Card, CardHeader, CardFooter, CardTitle, CardDescription,CardContent }
```

components/ui/checkbox.tsx

```
import * as React from "react"
import * as CheckboxPrimitive from "@radix-ui/react-checkbox"
import { Check } from "lucide-react"

import { cn } from "@/lib/utils"

const Checkbox = React.forwardRef<
  React.ElementRef<typeof CheckboxPrimitive.Root>,
  React.ComponentPropsWithoutRef<typeof CheckboxPrimitive.Root>
>(({ className, ...props }, ref) =>
```

```
<CheckboxPrimitive.Root
  ref={ref}
  className={cn(
    "peer h-4 w-4 shrink-0 rounded-sm border border-primary shadow focus-visible:outline-none focus-visible:ring-1 focus-visible:ring-ring disabled:cursor-not-allowed disabled:opacity-50 data-[state=checked]:bg-primary data-[state=checked]:text-primary-foreground",
    className
  )}
  {...props}
>
  <CheckboxPrimitive.Indicator
    className={cn("flex items-center justify-center text-current")}>
    <Check className="h-4 w-4" />
  </CheckboxPrimitive.Indicator>
</CheckboxPrimitive.Root>
))
Checkbox.displayName = CheckboxPrimitive.Root.displayName

export { Checkbox }
```

components/ui/input.tsx

```
import * as React from "react"

import { cn } from "@/lib/utils"

const Input = React.forwardRef<HTMLInputElement, React.ComponentProps<"input">>(
  ({ className, type, ...props }, ref) => {
    return (
      <input
        type={type}
        className={cn(
          "flex h-9 w-full rounded-md border border-input bg-transparent px-3 py-1 text-base shadow-sm transition-colors file:border-0 file:bg-transparent file:text-sm file:font-medium file:text-foreground placeholder:text-muted-foreground focus-visible:outline-none focus-visible:ring-1 focus-visible:ring-ring disabled:cursor-not-allowed disabled:opacity-50 md:text-sm",
          className
        )}
        ref={ref}
        {...props}
      />
    )
  }
)
Input.displayName = "Input"

export { Input }
```

components/ui/label.tsx

```
import * as React from "react"
import * as LabelPrimitive from "@radix-ui/react-label"
import { cva, type VariantProps } from "class-variance-authority"

import { cn } from "@/lib/utils"

const labelVariants = cva(
  "text-sm font-medium leading-none peer-disabled:cursor-not-allowed peer-
disabled:opacity-70"
)

const Label = React.forwardRef<
  React.ElementRef<typeof LabelPrimitive.Root>,
  React.ComponentPropsWithoutRef<typeof LabelPrimitive.Root> &
  VariantProps<typeof labelVariants>
>(({ className, ...props }, ref) => (
  <LabelPrimitive.Root
    ref={ref}
    className={cn(labelVariants(), className)}
    {...props}
  />
))
Label.displayName = LabelPrimitive.Root.displayName

export { Label }
```

components/ui/select.tsx

```
import * as React from "react"
import * as LabelPrimitive from "@radix-ui/react-label"
import { cva, type VariantProps } from "class-variance-authority"

import { cn } from "@/lib/utils"

const labelVariants = cva(
  "text-sm font-medium leading-none peer-disabled:cursor-not-allowed peer-
disabled:opacity-70"
)

const Label = React.forwardRef<
  React.ElementRef<typeof LabelPrimitive.Root>,
  React.ComponentPropsWithoutRef<typeof LabelPrimitive.Root> &
  VariantProps<typeof labelVariants>
>(({ className, ...props }, ref) => (
  <LabelPrimitive.Root
    ref={ref}
    className={cn(labelVariants(), className)}
    {...props}
  />
))
Label.displayName = LabelPrimitive.Root.displayName

export { Label }
```

```
    />
  ))
Label.displayName = LabelPrimitive.Root.displayName

export { Label }
```

components/ui/tabs.tsx

```
"use client"

import * as React from "react"
import * as TabsPrimitive from "@radix-ui/react-tabs"

import { cn } from "@/lib/utils"

const Tabs = TabsPrimitive.Root

const TabsList = React.forwardRef<
  React.ElementRef<typeof TabsPrimitive.List>,
  React.ComponentPropsWithoutRef<typeof TabsPrimitive.List>
>(({ className, ...props }, ref) => (
  <TabsPrimitive.List
    ref={ref}
    className={cn(
      "inline-flex h-9 items-center justify-center rounded-lg bg-muted p-1 text-muted-foreground",
      className
    )}
    {...props}
  />
))
TabsList.displayName = TabsPrimitive.List.displayName

const TabsTrigger = React.forwardRef<
  React.ElementRef<typeof TabsPrimitive.Trigger>,
  React.ComponentPropsWithoutRef<typeof TabsPrimitive.Trigger>
>(({ className, ...props }, ref) => (
  <TabsPrimitive.Trigger
    ref={ref}
    className={cn(
      "inline-flex items-center justify-center whitespace nowrap rounded-md px-3 py-1 text-sm font-medium ring-offset-background transition-all focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2 disabled:pointer-events-none disabled:opacity-50 data-[state=active]:bg-background data-[state=active]:text-foreground data-[state=active]:shadow",
      className
    )}
    {...props}
  />
))
TabsTrigger.displayName = TabsPrimitive.Trigger.displayName
```

```
const TabsContent = React.forwardRef<
  React.ElementRef<typeof TabsPrimitive.Content>,
  React.ComponentPropsWithoutRef<typeof TabsPrimitive.Content>
>(({ className, ...props }, ref) => (
  <TabsPrimitive.Content
    ref={ref}
    className={cn(
      "mt-2 ring-offset-background focus-visible:outline-none focus-visible:ring-2
      focus-visible:ring-ring focus-visible:ring-offset-2",
      className
    )}
    {...props}
  />
))
TabsContent.displayName = TabsPrimitive.Content.displayName

export { Tabs, TabsList, TabsTrigger, TabsContent }
```

components/ui/textarea.tsx

```
import * as React from "react"

import { cn } from "@/lib/utils"

const Textarea = React.forwardRef<
  HTMLTextAreaElement,
  React.ComponentProps<"textarea">
>(({ className, ...props }, ref) => {
  return (
    <textarea
      className={cn(
        "flex min-h-[60px] w-full rounded-md border border-input bg-transparent
        px-3 py-2 text-base shadow-sm placeholder:text-muted-foreground focus-
        visible:outline-none focus-visible:ring-1 focus-visible:ring-ring disabled:cursor-
        not-allowed disabled:opacity-50 md:text-sm",
        className
      )}
      ref={ref}
      {...props}
    />
  )
})
Textarea.displayName = "Textarea"

export { Textarea }
```

lib/utils.ts

```
import { clsx, type ClassValue } from "clsx";
import { twMerge } from "tailwind-merge";

export function cn(...inputs: ClassValue[]) {
  return twMerge(clsx(inputs));
}
```

App.tsx

```
import { useState } from 'react'
import { Card,CardContent, CardHeader, CardTitle } from '@/components/ui/card'
import { Input } from '@/components/ui/input'
import { Label } from '@/components/ui/label'
import { Button } from '@/components/ui/button'
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from
'@/components/ui/select'
import { Textarea } from '@/components/ui/textarea'
import { Checkbox } from '@/components/ui/checkbox'
import { Tabs, TabsContent, TabsList, TabsTrigger } from '@/components/ui/tabs'

function App() {
  const [formData, setFormData] = useState({
    // Location Details
    locationName: 'Central Distribution Center',
    customerCode: 'REQ',
    customerName: '07B Request Company - USA Details Land UNITS',
    locationCode: 'CDC-001',
    locationStatus: 'active',

    // Facility
    facilityType: 'warehouse',

    // Address
    address: '1234 Industrial Parkway\\nSuite 100',
    city: 'Los Angeles',
    state: 'CA',
    zipCode: '90210',
    country: 'UNITED STATES',

    // Contact
    contactName: 'John Smith',
    phoneNumber: '(555) 123-4567',
    emailAddress: 'john.smith@requestcompany.com',

    // Comments
    comments: 'This is the primary distribution center for the West Coast
operations. 24/7 operations with loading dock access.'
  })

  const handleInputChange = (field: string, value: string) => {
    setFormData(prev => ({ ...prev, [field]: value }))
}
```

```
}

return (
  <div className="min-h-screen bg-gray-50">
    {/* Header */}
    <div className="bg-white border-b">
      <div className="max-w-7xl mx-auto px-4 py-4">
        <div className="flex items-center justify-between">
          <div className="flex items-center space-x-4">
            <h1 className="text-xl font-semibold text-gray-900">Location</h1>
            <div className="text-sm text-gray-500">
              Edit Location: 07B Request Company - USA Details Land UNITS
            </div>
          </div>
        </div>
        <div className="flex items-center space-x-2">
          <Button variant="outline" size="sm">Print</Button>
          <Button variant="outline" size="sm">Save</Button>
          <Button variant="outline" size="sm">Cancel</Button>
        </div>
      </div>
    </div>
  </div>

<div className="max-w-7xl mx-auto px-4 py-6">
  {/* Navigation Tabs */}
  <Tabs defaultValue="general" className="mb-6">
    <TabsList className="bg-white border">
      <TabsTrigger value="general">General</TabsTrigger>
      <TabsTrigger value="operational">Operational</TabsTrigger>
      <TabsTrigger value="financial">Financial</TabsTrigger>
      <TabsTrigger value="legal">Legal</TabsTrigger>
      <TabsTrigger value="environmental">Environmental</TabsTrigger>
      <TabsTrigger value="additional">Additional</TabsTrigger>
    </TabsList>

    <TabsContent value="general" className="space-y-6">
      {/* Location Details Section */}
      <Card>
        <CardHeader>
          <CardTitle className="text-lg font-medium">Location
Details</CardTitle>
        </CardHeader>
        <CardContent className="space-y-4">
          <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3
gap-4">
            <div className="space-y-2">
              <Label htmlFor="locationName">Location Name</Label>
              <Input
                id="locationName"
                value={formData.locationName}
                onChange={(e) => handleInputChange('locationName',
e.target.value)}
                placeholder="Enter location name"
              />
            </div>
          </div>
        </CardContent>
      </Card>
    </TabsContent>
  </Tabs>
</div>
```

```
</div>
<div className="space-y-2">
  <Label htmlFor="customerCode">Customer Code</Label>
  <Select value={formData.customerCode} onChange={(value)} => handleInputChange('customerCode', value)>
    <SelectTrigger>
      <SelectValue placeholder="Select customer code" />
    </SelectTrigger>
    <SelectContent>
      <SelectItem value="REQ">REQ Request Company</SelectItem>
      <SelectItem value="ABC">ABC Company</SelectItem>
      <SelectItem value="XYZ">XYZ Corp</SelectItem>
    </SelectContent>
  </Select>
</div>
<div className="space-y-2">
  <Label htmlFor="customerName">Customer Name</Label>
  <Input
    id="customerName"
    value={formData.customerName}
    onChange={(e) => handleInputChange('customerName', e.target.value)}
    placeholder="07B Request Company - USA Details Land UNITS"
  />
</div>
<div className="space-y-2">
  <Label htmlFor="locationCode">Location Code</Label>
  <Input
    id="locationCode"
    value={formData.locationCode}
    onChange={(e) => handleInputChange('locationCode', e.target.value)}
    placeholder="Enter location code"
  />
</div>
<div className="space-y-2">
  <Label htmlFor="locationStatus">Location Status</Label>
  <Select value={formData.locationStatus} onChange={(value)} => handleInputChange('locationStatus', value)>
    <SelectTrigger>
      <SelectValue placeholder="Select status" />
    </SelectTrigger>
    <SelectContent>
      <SelectItem value="active">Active</SelectItem>
      <SelectItem value="inactive">Inactive</SelectItem>
      <SelectItem value="pending">Pending</SelectItem>
    </SelectContent>
  </Select>
</div>
</CardContent>
</Card>

/* Facility Section */
```

```
<Card>
  <CardHeader>
    <CardTitle className="text-lg font-medium">Facility</CardTitle>
  </CardHeader>
  <CardContent>
    <div className="space-y-2">
      <Label htmlFor="facilityType">Facility Type</Label>
      <Select value={formData.facilityType} onChange={(value) =>
handleInputChange('facilityType', value)}>
        <SelectTrigger className="w-full md:w-1/3">
          <SelectValue placeholder="Select facility type" />
        </SelectTrigger>
        <SelectContent>
          <SelectItem value="warehouse">Warehouse</SelectItem>
          <SelectItem value="office">Office</SelectItem>
          <SelectItem value="manufacturing">Manufacturing</SelectItem>
          <SelectItem value="retail">Retail</SelectItem>
        </SelectContent>
      </Select>
    </div>
  </CardContent>
</Card>

{/* Address Section */}
<Card>
  <CardHeader>
    <CardTitle className="text-lg font-medium">Address</CardTitle>
  </CardHeader>
  <CardContent>
    <div className="grid grid-cols-1 lg:grid-cols-2 gap-6">
      <div className="space-y-4">
        <div className="space-y-2">
          <Label htmlFor="address">Address</Label>
          <Textarea
            id="address"
            value={formData.address}
            onChange={(e) => handleInputChange('address',
e.target.value)}
            placeholder="Enter street address"
            rows={3}
          />
        </div>
        <div className="grid grid-cols-2 gap-4">
          <div className="space-y-2">
            <Label htmlFor="city">City</Label>
            <Input
              id="city"
              value={formData.city}
              onChange={(e) => handleInputChange('city',
e.target.value)}
              placeholder="Enter city"
            />
          </div>
        <div className="space-y-2">

```

```
<Label htmlFor="state">State</Label>
<Select value={formData.state} onChange={(value) =>
handleInputChange('state', value)}>
  <SelectTrigger>
    <SelectValue placeholder="Select state" />
  </SelectTrigger>
  <SelectContent>
    <SelectItem value="CA">California</SelectItem>
    <SelectItem value="NY">New York</SelectItem>
    <SelectItem value="TX">Texas</SelectItem>
    <SelectItem value="FL">Florida</SelectItem>
  </SelectContent>
</Select>
</div>
</div>
<div className="grid grid-cols-2 gap-4">
  <div className="space-y-2">
    <Label htmlFor="zipCode">Zip Code</Label>
    <Input
      id="zipCode"
      value={formData.zipCode}
      onChange={(e) => handleInputChange('zipCode',
e.target.value)}
      placeholder="Enter zip code"
    />
  </div>
  <div className="space-y-2">
    <Label htmlFor="country">Country</Label>
    <Select value={formData.country} onChange={(value) =>
handleInputChange('country', value)}>
      <SelectTrigger>
        <SelectValue />
      </SelectTrigger>
      <SelectContent>
        <SelectItem value="UNITED STATES">UNITED
        STATES</SelectItem>
        <SelectItem value="CANADA">CANADA</SelectItem>
        <SelectItem value="MEXICO">MEXICO</SelectItem>
      </SelectContent>
    </Select>
  </div>
</div>
<div className="bg-gray-100 rounded-lg p-4 flex flex-col items-center justify-center min-h-[300px] relative overflow-hidden">
  {/* Mock map with grid pattern */}
  <div className="absolute inset-0 opacity-10">
    <div className="grid grid-cols-8 grid-rows-6 h-full w-full">
      {Array.from({ length: 48 }).map((_, i) => (
        <div key={`grid-${i}`}>
          <div style={{ border: '1px solid #ccc', width: 12.5, height: 12.5 }></div>
        </div>
      ))}
    </div>
  </div>
</div>
```

```
    {/* Roads pattern */}
    <div className="absolute inset-0">
        <div className="absolute top-1/3 left-0 right-0 h-1 bg-gray-400 opacity-30" />
        <div className="absolute top-2/3 left-0 right-0 h-1 bg-gray-400 opacity-30" />
            <div className="absolute left-1/3 top-0 bottom-0 w-1 bg-gray-400 opacity-30" />
            <div className="absolute left-2/3 top-0 bottom-0 w-1 bg-gray-400 opacity-30" />
        </div>

    {/* Location marker */}
    <div className="relative z-10 text-center">
        <div className="w-12 h-12 mx-auto bg-red-500 rounded-full flex items-center justify-center mb-2 shadow-lg">
            <svg className="w-6 h-6 text-white" fill="currentColor" viewBox="0 0 24 24">
                <path d="M12 2C8.13 2 5 5.13 5 9c0 5.25 7 13 7 13s7-7.75 7-13c0-3.87-3.13-7-7-7zm0 9.5c-1.38 0-2.5-1.12-2.5-2.5s1.12-2.5 2.5-2.5 2.5 1.12 2.5 2.5-1.12 2.5-2.5 2.5z"/>
            </svg>
        </div>
        <p className="font-medium text-gray-700">Map View</p>
        <p className="text-sm text-gray-500">Los Angeles, CA 90210</p>
        <div className="mt-2 text-xs text-gray-400">
            <div className="flex items-center justify-center space-x-2">
                <span>Lat: 34.0522</span>
                <span>•</span>
                <span>Lng: -118.2437</span>
            </div>
        </div>
    </div>
</CardContent>
</Card>

    {/* Contact Section */}
<Card>
    <CardHeader>
        <CardTitle className="text-lg font-medium">Contact</CardTitle>
    </CardHeader>
    <CardContent className="space-y-4">
        <div className="grid grid-cols-1 md:grid-cols-3 gap-4">
            <div className="space-y-2">
                <Label htmlFor="contactName">Contact Name</Label>
                <Input
                    id="contactName"
                    value={formData.contactName}
                    onChange={(e) => handleInputChange('contactName',
```

```
e.target.value)}  
        placeholder="Enter contact name"  
    />  
  </div>  
  <div className="space-y-2">  
    <Label htmlFor="phoneNumber">Phone Number</Label>  
    <Input  
      id="phoneNumber"  
      type="tel"  
      value={formData.phoneNumber}  
      onChange={(e) => handleInputChange('phoneNumber',  
e.target.value)}  
        placeholder="Enter phone number"  
    />  
  </div>  
  <div className="space-y-2">  
    <Label htmlFor="emailAddress">Email Address</Label>  
    <Input  
      id="emailAddress"  
      type="email"  
      value={formData.emailAddress}  
      onChange={(e) => handleInputChange('emailAddress',  
e.target.value)}  
        placeholder="Enter email address"  
    />  
  </div>  
  </CardContent>  
</Card>  
  
 {/* Comments Section */}  
<Card>  
  <CardHeader>  
    <CardTitle className="text-lg font-medium">Comments</CardTitle>  
  </CardHeader>  
  <CardContent>  
    <div className="space-y-2">  
      <Label htmlFor="comments">Comments</Label>  
      <Textarea  
        id="comments"  
        value={formData.comments}  
        onChange={(e) => handleInputChange('comments',  
e.target.value)}  
          placeholder="Enter any additional comments or notes"  
        rows={4}  
      />  
    </div>  
  </CardContent>  
</Card>  
</TabsContent>  
  
 {/* Other tab contents */}  
<TabsContent value="operational">  
  <Card>
```

```
<CardHeader>
  <CardTitle>Operational Information</CardTitle>
</CardHeader>
<CardContent>
  <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
    <div className="space-y-4">
      <div className="space-y-2">
        <Label htmlFor="operatingHours">Operating Hours</Label>
        <Input placeholder="24/7" />
      </div>
      <div className="space-y-2">
        <Label htmlFor="capacity">Storage Capacity</Label>
        <Input placeholder="50,000 sq ft" />
      </div>
      <div className="space-y-2">
        <Label htmlFor="loadingDocks">Loading Docks</Label>
        <Input placeholder="12 dock doors" />
      </div>
    </div>
    <div className="space-y-4">
      <div className="space-y-2">
        <Label htmlFor="securityLevel">Security Level</Label>
        <Select>
          <SelectTrigger>
            <SelectValue placeholder="Select security level" />
          </SelectTrigger>
          <SelectContent>
            <SelectItem value="basic">Basic</SelectItem>
            <SelectItem value="standard">Standard</SelectItem>
            <SelectItem value="high">High Security</SelectItem>
          </SelectContent>
        </Select>
      </div>
      <div className="space-y-2">
        <Label htmlFor="staffCount">Staff Count</Label>
        <Input placeholder="25 employees" />
      </div>
    </div>
  </div>
</CardContent>
</Card>
</TabsContent>

<TabsContent value="financial">
<Card>
  <CardHeader>
    <CardTitle>Financial Information</CardTitle>
  </CardHeader>
  <CardContent>
    <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
      <div className="space-y-4">
        <div className="space-y-2">
          <Label htmlFor="monthlyRent">Monthly Rent</Label>
          <Input placeholder="$25,000" />
        </div>
      </div>
    </div>
  </CardContent>
</Card>
</TabsContent>
```

```
</div>
<div className="space-y-2">
  <Label htmlFor="deposit">Security Deposit</Label>
  <Input placeholder="$50,000" />
</div>
</div>
<div className="space-y-4">
  <div className="space-y-2">
    <Label htmlFor="leaseStart">Lease Start Date</Label>
    <Input type="date" />
  </div>
  <div className="space-y-2">
    <Label htmlFor="leaseEnd">Lease End Date</Label>
    <Input type="date" />
  </div>
</div>
</div>
</CardContent>
</Card>
</TabsContent>

<TabsContent value="legal">
  <Card>
    <CardHeader>
      <CardTitle>Legal Information</CardTitle>
    </CardHeader>
    <CardContent>
      <div className="space-y-4">
        <div className="space-y-2">
          <Label htmlFor="permits">Required Permits</Label>
          <Textarea placeholder="List all required permits and licenses" rows={3} />
        </div>
        <div className="space-y-2">
          <Label htmlFor="compliance">Compliance Requirements</Label>
          <Textarea placeholder="Describe compliance requirements" rows={3} />
        </div>
      </div>
    </CardContent>
  </Card>
</TabsContent>

<TabsContent value="environmental">
  <Card>
    <CardHeader>
      <CardTitle>Environmental Information</CardTitle>
    </CardHeader>
    <CardContent>
      <div className="space-y-4">
        <div className="space-y-2">
          <Label htmlFor="hazmat">Hazardous Materials</Label>
          <Select>
            <SelectTrigger>
```

```
        <SelectValue placeholder="Select hazmat classification" />
        </SelectTrigger>
        <SelectContent>
            <SelectItem value="none">None</SelectItem>
            <SelectItem value="class1">Class 1 -
Explosives</SelectItem>
            <SelectItem value="class2">Class 2 - Gases</SelectItem>
            <SelectItem value="class3">Class 3 - Flammable
Liquids</SelectItem>
        </SelectContent>
    </Select>
</div>
<div className="space-y-2">
    <Label htmlFor="environmental">Environmental
Compliance</Label>
    <Textarea placeholder="Environmental regulations and compliance notes" rows={3} />
</div>
</div>
</CardContent>
</Card>
</TabsContent>

<TabsContent value="additional">
<Card>
    <CardHeader>
        <CardTitle>Additional Information</CardTitle>
    </CardHeader>
    <CardContent>
        <div className="space-y-4">
            <div className="space-y-2">
                <Label htmlFor="notes">Additional Notes</Label>
                <Textarea placeholder="Any additional information or special requirements" rows={4} />
            </div>
            <div className="space-y-2">
                <Label htmlFor="attachments">Attachments</Label>
                <div className="border-2 border-dashed border-gray-300 rounded-lg p-6 text-center">
                    <img alt="Upload attachments icon" data-bbox="311 708 896 808" style="margin: auto; width: 100%; height: 100%;"/>
                    <div data-bbox="311 811 896 896">
                        <Button variant="outline">Upload Files</Button>
                    </div>
                    <p data-bbox="311 898 896 918">PNG, JPG, PDF up to 10MB</p>
                </div>
            </div>
        </div>
    </CardContent>
</Card>
</TabsContent>
```

```
        </CardContent>
    </Card>
</TabsContent>
</Tabs>

{/* Action Buttons */}


<Button variant="outline">Cancel</Button>
    <Button variant="outline">Save Draft</Button>
    <Button>Save & Submit</Button>


</div>
</div>
</div>
)

}

export default App
```

Cursor

EditForm.tsx

```
// 12:15:app/src/EditForm.tsx
import React, { useState } from 'react';

const EditForm: React.FC = () => {
    const [formData, setFormData] = useState({
        name: '',
        address: '',
        locality: '',
        region: '',
        contacts: [{ firstName: '', email: '' }],
        comments: '',
    });

    const handleChange = (e: React.ChangeEvent<HTMLInputElement | HTMLTextAreaElement>, index?: number) => {
        const { name, value } = e.target;

        if (index !== undefined) {
            const updatedContacts = [...formData.contacts];
            updatedContacts[index] = { ...updatedContacts[index], [name]: value };
            setFormData({ ...formData, contacts: updatedContacts });
        } else {
            setFormData({ ...formData, [name]: value });
        }
    };

    const addContact = () => {
        setFormData({ ...formData, contacts: [...formData.contacts, { firstName: '', email: '' }] });
    };
}
```

```
};

const handleSubmit = (e: React.FormEvent) => {
  e.preventDefault();
  console.log('Form submitted:', formData);
  // Here you can add the API call to save the data
};

return (
  <form onSubmit={handleSubmit}>
    <h1>Edit Location</h1>
    <div>
      <label>Name:</label>
      <input type="text" name="name" value={formData.name} onChange={handleChange} required />
    </div>
    <div>
      <label>Address:</label>
      <input type="text" name="address" value={formData.address} onChange={handleChange} required />
    </div>
    <div>
      <label>Locality:</label>
      <input type="text" name="locality" value={formData.locality} onChange={handleChange} required />
    </div>
    <div>
      <label>Region:</label>
      <input type="text" name="region" value={formData.region} onChange={handleChange} required />
    </div>
    <h2>Contacts</h2>
    {formData.contacts.map((contact, index) => (
      <div key={index}>
        <label>First Name:</label>
        <input
          type="text"
          name="firstName"
          value={contact.firstName}
          onChange={(e) => handleChange(e, index)}
          required
        />
        <label>Email:</label>
        <input
          type="email"
          name="email"
          value={contact.email}
          onChange={(e) => handleChange(e, index)}
          required
        />
      </div>
    ))}
    <button type="button" onClick={addContact}>Add Contact</button>
  </form>
)
```

```
        <label>Comments:</label>
        <textarea name="comments" value={formData.comments} onChange={handleChange} />
    </div>
    <button type="submit">Save</button>
</form>
);
};

export default EditForm;
```

App.tsx

```
import React from 'react';
import EditForm from './EditForm';

const App: React.FC = () => {
    return (
        <div>
            <EditForm />
        </div>
    );
};

export default App;
```

App.css

```
/* 12:15:app/src/App.css */
form {
    max-width: 600px;
    margin: auto;
    padding: 20px;
    border: 1px solid #ccc;
    border-radius: 5px;
}

div {
    margin-bottom: 15px;
}

label {
    display: block;
    margin-bottom: 5px;
}

input, textarea {
    width: 100%;
    padding: 8px;
```

```
border: 1px solid #ccc;
border-radius: 4px;
}

button {
  padding: 10px 15px;
  background-color: #007bff;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}

button:hover {
  background-color: #0056b3;
}
```

Claude Opus 4

EditPage.tsx

```
import React, { useState } from 'react';
import { ChevronDown, ChevronRight, X, Plus, Upload, Save, MapPin } from 'lucide-react';

const CompanyEditPage = () => {
  const [activeTab, setActiveTab] = useState('general');
  const [expandedSections, setExpandedSections] = useState({
    company: true,
    location: true,
    contacts: false,
    comments: false
  });

  const [formData, setFormData] = useState({
    company: {
      id: 'ACME123',
      name: 'All Tenant Company - USA, Mexia Lake (B817)',
      type: 'SELECT COMPANY',
      number: '123456',
      status: 'Active'
    },
    location: {
      name: 'Mexia Lake',
      code: 'ML',
      country: 'UNITED STATES',
      lat: '31.6819',
      lon: '-96.4828'
    }
  });
}
```

```
const toggleSection = (section) => {
  setExpandedSections(prev => ({
    ...prev,
    [section]: !prev[section]
  }));
};

const handleInputChange = (section, field, value) => {
  setFormData(prev => ({
    ...prev,
    [section]: {
      ...prev[section],
      [field]: value
    }
  }));
};

const SectionHeader = ({ title, isExpanded, onToggle }) => (
  <div
    className="section-header"
    onClick={onToggle}
  >
    {isExpanded ? <ChevronDown size={16} /> : <ChevronRight size={16} />}
    <span>{title}</span>
  </div>
);

return (
  <div className="edit-page">
    <style jsx>{
      .edit-page {
        font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto,
        sans-serif;
        font-size: 13px;
        color: #333;
        background-color: #f5f5f5;
        min-height: 100vh;
      }

      .header {
        background-color: #7cb342;
        color: white;
        padding: 8px 16px;
        display: flex;
        align-items: center;
        justify-content: space-between;
      }

      .header-title {
        font-size: 14px;
        font-weight: 500;
      }
    </style>
  
```

```
.header-actions {  
    display: flex;  
    gap: 8px;  
}  
  
.breadcrumb {  
    background-color: white;  
    padding: 8px 16px;  
    border-bottom: 1px solid #e0e0e0;  
    font-size: 12px;  
    color: #666;  
}  
  
.breadcrumb a {  
    color: #1976d2;  
    text-decoration: none;  
    margin-right: 8px;  
}  
  
.main-content {  
    display: flex;  
    height: calc(100vh - 100px);  
}  
  
.sidebar {  
    width: 200px;  
    background-color: white;  
    border-right: 1px solid #e0e0e0;  
    overflow-y: auto;  
}  
  
.content-area {  
    flex: 1;  
    display: flex;  
    overflow: hidden;  
}  
  
.form-section {  
    flex: 1;  
    padding: 16px;  
    overflow-y: auto;  
    background-color: white;  
    margin: 8px;  
    border-radius: 4px;  
    box-shadow: 0 1px 3px rgba(0,0,0,0.1);  
}  
  
.map-section {  
    width: 400px;  
    padding: 16px;  
    background-color: white;  
    margin: 8px 8px 8px 0;  
    border-radius: 4px;  
    box-shadow: 0 1px 3px rgba(0,0,0,0.1);  
}
```

```
}

.tabs {
  display: flex;
  background-color: white;
  border-bottom: 1px solid #e0e0e0;
  padding: 0 16px;
}

.tab {
  padding: 12px 16px;
  cursor: pointer;
  border-bottom: 2px solid transparent;
  color: #666;
  transition: all 0.2s;
}

.tab:hover {
  color: #333;
}

.tab.active {
  color: #1976d2;
  border-bottom-color: #1976d2;
}

.section {
  margin-bottom: 24px;
}

.section-header {
  display: flex;
  align-items: center;
  gap: 8px;
  padding: 8px;
  background-color: #f5f5f5;
  cursor: pointer;
  user-select: none;
  font-weight: 500;
  border-radius: 4px;
  margin-bottom: 12px;
}

.section-header:hover {
  background-color: #eeeeee;
}

.section-content {
  padding: 0 8px;
}

.form-group {
  margin-bottom: 16px;
}
```

```
.form-row {  
  display: flex;  
  gap: 16px;  
  margin-bottom: 16px;  
}  
  
.form-field {  
  flex: 1;  
}  
  
label {  
  display: block;  
  margin-bottom: 4px;  
  color: #666;  
  font-size: 12px;  
}  
  
input[type="text"],  
input[type="number"],  
select {  
  width: 100%;  
  padding: 6px 8px;  
  border: 1px solid #d0d0d0;  
  border-radius: 3px;  
  font-size: 13px;  
}  
  
input[type="text"]:focus,  
input[type="number"]:focus,  
select:focus {  
  outline: none;  
  border-color: #1976d2;  
  box-shadow: 0 0 0 2px rgba(25, 118, 210, 0.2);  
}  
  
input[readonly] {  
  background-color: #f5f5f5;  
  cursor: not-allowed;  
}  
  
.button {  
  padding: 6px 12px;  
  border: 1px solid #d0d0d0;  
  background-color: white;  
  border-radius: 3px;  
  cursor: pointer;  
  font-size: 12px;  
  display: inline-flex;  
  align-items: center;  
  gap: 4px;  
  transition: all 0.2s;  
}
```

```
.button:hover {  
    background-color: #f5f5f5;  
}  
  
.button-primary {  
    background-color: #1976d2;  
    color: white;  
    border-color: #1976d2;  
}  
  
.button-primary:hover {  
    background-color: #1565c0;  
}  
  
.map-container {  
    height: 300px;  
    background-color: #e0e0e0;  
    border-radius: 4px;  
    display: flex;  
    align-items: center;  
    justify-content: center;  
    color: #666;  
    margin-top: 16px;  
}  
  
.status-active {  
    color: #4caf50;  
    font-weight: 500;  
}  
  
.sidebar-menu {  
    padding: 8px 0;  
}  
  
.sidebar-item {  
    padding: 8px 16px;  
    cursor: pointer;  
    transition: background-color 0.2s;  
}  
  
.sidebar-item:hover {  
    background-color: #f5f5f5;  
}  
  
.sidebar-item.active {  
    background-color: #e3f2fd;  
    color: #1976d2;  
    font-weight: 500;  
}  
  
.action-bar {  
    display: flex;  
    justify-content: flex-end;  
    gap: 8px;
```

```
padding: 16px;
border-top: 1px solid #e0e0e0;
background-color: #fafafa;
margin: -16px -16px 16px -16px;
}

`}</style>

{/* Header */}
<div className="header">
  <div className="header-title">
    <MapPin size={16} style={{ display: 'inline', marginRight: '8px' }} />
    Location
  </div>
  <div className="header-actions">
    <button className="button" style={{ backgroundColor: 'transparent',
border: 'none', color: 'white' }}>
      <X size={16} />
    </button>
  </div>
</div>

{/* Breadcrumb */}
<div className="breadcrumb">
  <a href="#">Home</a> /
  <a href="#"> All Tenant Company - USA, Mexia Lake (B817)</a>
</div>

{/* Tabs */}
<div className="tabs">
  <div className={`tab ${activeTab === 'general' ? 'active' : ''}`}>
    onClick={() => setActiveTab('general')}
    General
  </div>
  <div className={`tab ${activeTab === 'additional' ? 'active' : ''}`}>
    onClick={() => setActiveTab('additional')}
    Additional Information
  </div>
</div>

{/* Main Content */}
<div className="main-content">
  {/* Sidebar */}
  <div className="sidebar">
    <div className="sidebar-menu">
      <div className="sidebar-item active">General Information</div>
      <div className="sidebar-item">Related Items</div>
      <div className="sidebar-item">Documents</div>
      <div className="sidebar-item">History</div>
    </div>
  </div>

  {/* Content Area */}
  <div className="content-area">
    {/* Form Section */}
```

```
<div className="form-section">
  /* Action Bar */
  <div className="action-bar">
    <button className="button">
      <X size={16} />
      Cancel
    </button>
    <button className="button button-primary">
      <Save size={16} />
      Save
    </button>
  </div>

  /* Company Section */
  <div className="section">
    <SectionHeader
      title="Company Details"
      isExpanded={expandedSections.company}
      onToggle={() => toggleSection('company')}
    />
    {expandedSections.company && (
      <div className="section-content">
        <div className="form-row">
          <div className="form-field">
            <label>ID</label>
            <input
              type="text"
              value={formData.company.id}
              readOnly
            />
          </div>
          <div className="form-field">
            <label>Type</label>
            <select
              value={formData.company.type}
              onChange={(e) => handleInputChange('company', 'type',
e.target.value)}
            >
              <option>SELECT COMPANY</option>
              <option>Corporation</option>
              <option>LLC</option>
              <option>Partnership</option>
            </select>
          </div>
        </div>
        <div className="form-group">
          <label>Company Name</label>
          <input
            type="text"
            value={formData.company.name}
            onChange={(e) => handleInputChange('company', 'name',
e.target.value)}
          />
        </div>
      </div>
    )}
  </div>
```

```
<div className="form-row">
  <div className="form-field">
    <label>Company Number</label>
    <input
      type="text"
      value={formData.company.number}
      onChange={(e) => handleInputChange('company', 'number',
e.target.value)}
    />
  </div>
  <div className="form-field">
    <label>Status</label>
    <div className="status-active" style={{ padding: '6px 0' }}>
      {formData.company.status}
    </div>
  </div>
</div>
)
</div>

/* Location Section */
<div className="section">
  <SectionHeader
    title="Location"
    isExpanded={expandedSections.location}
    onToggle={() => toggleSection('location')}
  />
  {expandedSections.location && (
    <div className="section-content">
      <div className="form-row">
        <div className="form-field">
          <label>Location Name</label>
          <input
            type="text"
            value={formData.location.name}
            onChange={(e) => handleInputChange('location', 'name',
e.target.value)}
          />
        </div>
        <div className="form-field" style={{ maxWidth: '100px' }}>
          <label>Code</label>
          <input
            type="text"
            value={formData.location.code}
            onChange={(e) => handleInputChange('location', 'code',
e.target.value)}
          />
        </div>
      </div>
      <div className="form-group">
        <label>Country</label>
        <select
          value={formData.location.country}
        </select>
      </div>
    </div>
  )
}
```

```
        onChange={(e) => handleInputChange('location', 'country',
e.target.value)}
      >
        <option>UNITED STATES</option>
        <option>CANADA</option>
        <option>MEXICO</option>
      </select>
    </div>
    <div className="form-row">
      <div className="form-field">
        <label>Latitude</label>
        <input
          type="number"
          value={formData.location.lat}
          onChange={(e) => handleInputChange('location', 'lat',
e.target.value)}
          step="0.0001"
        />
      </div>
      <div className="form-field">
        <label>Longitude</label>
        <input
          type="number"
          value={formData.location.lon}
          onChange={(e) => handleInputChange('location', 'lon',
e.target.value)}
          step="0.0001"
        />
      </div>
    </div>
  )}

/* Contacts Section */
<div className="section">
  <SectionHeader
    title="Contacts"
    isExpanded={expandedSections.contacts}
    onToggle={() => toggleSection('contacts')}
  />
  {expandedSections.contacts && (
    <div className="section-content">
      <button className="button">
        <Plus size={16} />
        Add Contact
      </button>
    </div>
  )}
</div>

/* Comments Section */
<div className="section">
  <SectionHeader>
```

```
        title="Comments"
        isExpanded={expandedSections.comments}
        onToggle={() => toggleSection('comments')}
      />
    {expandedSections.comments && (
      <div className="section-content">
        <textarea
          style={{
            width: '100%',
            minHeight: '100px',
            padding: '8px',
            border: '1px solid #d0d0d0',
            borderRadius: '3px',
            fontSize: '13px',
            resize: 'vertical'
          }}
          placeholder="Enter comments..."/>
      </div>
    )}
  </div>
</div>

/* Map Section */
<div className="map-section">
  <h3 style={{ marginTop: 0, marginBottom: '16px', fontSize: '16px' }}>Location Map</h3>
  <div style={{ marginBottom: '16px' }}>
    <p style={{ margin: '4px 0', fontSize: '13px', color: '#666' }}>
      <strong>Coordinates:</strong> {formData.location.lat},
    {formData.location.lon}
    </p>
    <p style={{ margin: '4px 0', fontSize: '13px', color: '#666' }}>
      <strong>Location:</strong> {formData.location.name},
    {formData.location.country}
    </p>
  </div>
  <div className="map-container">
    <div style={{ textAlign: 'center' }}>
      <MapPin size={48} color="#999" />
      <p>Map View</p>
      <p style={{ fontSize: '12px' }}>Latitude: {formData.location.lat}</p>
    </div>
    <p style={{ fontSize: '12px' }}>Longitude: {formData.location.lon}</p>
  </div>
  <div style={{ marginTop: '16px', display: 'flex', gap: '8px' }}>
    <button className="button" style={{ flex: 1 }}>
      <MapPin size={16} />
      Update Location
    </button>
    <button className="button">
      <Upload size={16} />
    </button>
  </div>
</div>
```

```
        </button>
      </div>
    </div>
  </div>
</div>
);
};

export default CompanyEditPage;
```

Claude Sonnet 4

EditPage.tsx

```
import React, { useState } from 'react';

// Types
type FormData = {
  // Location Details
  location: string;
  locationDetails: string;

  // Facility
  facility: string;

  // Address
  address: string;
  city: string;
  state: string;
  country: string;
  postalCode: string;
  region: string;

  // Contact
  contactName: string;
  contactTitle: string;
  phone: string;
  email: string;

  // Comments
  comments: string;
};

const EditPageForm: React.FC = () => {
  const [formData, setFormData] = useState<FormData>({
    location: 'JFK Airport Complex - JFK1, Room Lane USR15',
    locationDetails: 'J.F. 30, 08-17 - LOC/ROOM',
    facility: '',
    address: '2707 South Sherborn',
    city: 'Bronx LIND',
  });
}
```

```
state: '10024',
country: 'UNITED STATES',
postalCode: '',
region: '',
contactName: '',
contactTitle: '',
phone: '',
email: '',
comments: ''
});

const [isExpanded, setIsExpanded] = useState({
location: true,
facility: true,
address: true,
contact: true,
comments: true
});

const handleInputChange = (field: keyof FormData, value: string) => {
setFormData(prev => ({
...prev,
[field]: value
}));
};

const toggleSection = (section: keyof typeof isExpanded) => {
setIsExpanded(prev => ({
...prev,
[section]: !prev[section]
}));
};

// Styles
const containerStyle: React.CSSProperties = {
fontFamily: 'Segoe UI, Tahoma, Geneva, Verdana, sans-serif',
backgroundColor: '#f5f5f5',
minHeight: '100vh',
padding: '0'
};

const headerStyle: React.CSSProperties = {
backgroundColor: '#4a6741',
color: 'white',
padding: '8px 16px',
fontSize: '14px',
fontWeight: 'normal',
margin: '0',
borderBottom: '1px solid #3a5431'
};

const sectionStyle: React.CSSProperties = {
backgroundColor: 'white',
border: '1px solid #d4d4d4',
```

```
        marginBottom: '2px'
    };

const sectionHeaderStyle: React.CSSProperties = {
    backgroundColor: '#e8e8e8',
    padding: '6px 12px',
    fontSize: '12px',
    fontWeight: '600',
    borderBottom: '1px solid #d4d4d4',
    cursor: 'pointer',
    display: 'flex',
    alignItems: 'center',
    gap: '6px'
};

const sectionContentStyle: React.CSSProperties = {
    padding: '12px',
    display: 'grid',
    gridTemplateColumns: 'repeat(auto-fit, minmax(200px, 1fr))',
    gap: '12px'
};

const fieldGroupStyle: React.CSSProperties = {
    display: 'flex',
    flexDirection: 'column',
    gap: '4px'
};

const labelStyle: React.CSSProperties = {
    fontSize: '11px',
    fontWeight: '600',
    color: '#333',
    textTransform: 'uppercase'
};

const inputStyle: React.CSSProperties = {
    border: '1px solid #ccc',
    padding: '4px 6px',
    fontSize: '12px',
    backgroundColor: 'white'
};

const selectStyle: React.CSSProperties = {
    border: '1px solid #ccc',
    padding: '4px 6px',
    fontSize: '12px',
    backgroundColor: 'white'
};

const textAreaStyle: React.CSSProperties = {
    border: '1px solid #ccc',
    padding: '6px',
    fontSize: '12px',
    backgroundColor: 'white',
```

```
minHeight: '60px',
resize: 'vertical'
};

const mapContainerStyle: React.CSSProperties = {
gridColumn: 'span 2',
height: '200px',
backgroundColor: '#f0f0f0',
border: '1px solid #ccc',
display: 'flex',
alignItems: 'center',
justifyContent: 'center',
fontSize: '14px',
color: '#666',
position: 'relative'
};

const expandIcon = (expanded: boolean) => expanded ? '▼' : '▶';

return (
<div style={containerStyle}>
  {/* Header */}
  <div style={headerStyle}>
    Location
  </div>

  {/* Location Details Section */}
  <div style={sectionStyle}>
    <div
      style={sectionHeaderStyle}
      onClick={() => toggleSection('location')}
    >
      <span>{expandIcon(isExpanded.location)}</span>
      <span>Location Details</span>
    </div>
    {isExpanded.location && (
      <div style={sectionContentStyle}>
        <div style={fieldGroupStyle}>
          <label style={labelStyle}>Location</label>
          <input
            style={inputStyle}
            value={formData.location}
            onChange={(e) => handleInputChange('location', e.target.value)}
          />
        </div>
        <div style={fieldGroupStyle}>
          <label style={labelStyle}>Location Details</label>
          <input
            style={inputStyle}
            value={formData.locationDetails}
            onChange={(e) => handleInputChange('locationDetails',
e.target.value)}
          />
        </div>
      </div>
    )}
  </div>
)
```

```
        </div>
    )}
</div>

{/* Facility Section */}
<div style={sectionStyle}>
  <div
    style={sectionHeaderStyle}
    onClick={() => toggleSection('facility')}
  >
    <span>{expandIcon(isExpanded.facility)}</span>
    <span>Facility</span>
  </div>
  {isExpanded.facility && (
    <div style={sectionContentStyle}>
      <div style={fieldGroupStyle}>
        <label style={labelStyle}>Facility</label>
        <select
          style={selectStyle}
          value={formData.facility}
          onChange={(e) => handleInputChange('facility', e.target.value)}
        >
          <option value="">Select Facility</option>
          <option value="warehouse">Warehouse</option>
          <option value="office">Office</option>
          <option value="distribution">Distribution Center</option>
        </select>
      </div>
    </div>
  )}
</div>

{/* Address Section */}
<div style={sectionStyle}>
  <div
    style={sectionHeaderStyle}
    onClick={() => toggleSection('address')}
  >
    <span>{expandIcon(isExpanded.address)}</span>
    <span>Address</span>
  </div>
  {isExpanded.address && (
    <div style={sectionContentStyle}>
      <div style={fieldGroupStyle}>
        <label style={labelStyle}>Address</label>
        <input
          style={inputStyle}
          value={formData.address}
          onChange={(e) => handleInputChange('address', e.target.value)}
        />
      </div>
      <div style={fieldGroupStyle}>
        <label style={labelStyle}>City</label>
        <input
```

```
        style={inputStyle}
        value={formData.city}
        onChange={(e) => handleInputChange('city', e.target.value)}
      />
    </div>
    <div style={fieldGroupStyle}>
      <label style={labelStyle}>State</label>
      <input
        style={inputStyle}
        value={formData.state}
        onChange={(e) => handleInputChange('state', e.target.value)}
      />
    </div>
    <div style={fieldGroupStyle}>
      <label style={labelStyle}>Country</label>
      <select
        style={selectStyle}
        value={formData.country}
        onChange={(e) => handleInputChange('country', e.target.value)}
      >
        <option value="">Select Country</option>
        <option value="UNITED STATES">United States</option>
        <option value="CANADA">Canada</option>
        <option value="MEXICO">Mexico</option>
      </select>
    </div>
    <div style={fieldGroupStyle}>
      <label style={labelStyle}>Postal Code</label>
      <input
        style={inputStyle}
        value={formData.postalCode}
        onChange={(e) => handleInputChange('postalCode', e.target.value)}
      />
    </div>
    <div style={fieldGroupStyle}>
      <label style={labelStyle}>Region</label>
      <select
        style={selectStyle}
        value={formData.region}
        onChange={(e) => handleInputChange('region', e.target.value)}
      >
        <option value="">Select Region</option>
        <option value="NORTHEAST">Northeast</option>
        <option value="SOUTHEAST">Southeast</option>
        <option value="MIDWEST">Midwest</option>
        <option value="SOUTHWEST">Southwest</option>
        <option value="WEST">West</option>
      </select>
    </div>

    /* Map Component */
    <div style={mapContainerStyle}>
      <div style={{ textAlign: 'center' }}>
        <div style={{ fontSize: '16px', marginBottom: '8px' }}>地图</div>
      </div>
    </div>
```

```
        <div>Interactive Map</div>
        <div style={{ fontSize: '11px', color: '#999', marginTop: '4px'
    }}>
            Map showing: {formData.address || 'No address specified'}
        </div>
        {/* Simulated map controls */}
        <div style={{
            position: 'absolute',
            top: '8px',
            right: '8px',
            display: 'flex',
            flexDirection: 'column',
            gap: '2px'
        }}>
            <button style={{
                width: '24px',
                height: '24px',
                border: '1px solid #ccc',
                backgroundColor: 'white',
                cursor: 'pointer',
                fontSize: '12px'
            }}>+</button>
            <button style={{
                width: '24px',
                height: '24px',
                border: '1px solid #ccc',
                backgroundColor: 'white',
                cursor: 'pointer',
                fontSize: '12px'
            }}>-</button>
        </div>
    </div>
    </div>
    </div>
)
</div>

/* Contact Section */
<div style={sectionStyle}>
    <div
        style={sectionHeaderStyle}
        onClick={() => toggleSection('contact')}
    >
        <span>{expandIcon(isExpanded.contact)}</span>
        <span>Contact</span>
    </div>
    {isExpanded.contact && (
        <div style={sectionContentStyle}>
            <div style={fieldGroupStyle}>
                <label style={labelStyle}>Contact Name</label>
                <input
                    style={inputStyle}
                    value={formData.contactName}
                    onChange={(e) => handleInputChange('contactName', e.target.value)}
                >
            </div>
        </div>
    )
}
```

```
        placeholder="Enter contact name"
      />
    </div>
    <div style={fieldGroupStyle}>
      <label style={labelStyle}>Contact Title</label>
      <input
        style={inputStyle}
        value={formData.contactTitle}
        onChange={(e) => handleInputChange('contactTitle',
e.target.value)}
        placeholder="Enter contact title"
      />
    </div>
    <div style={fieldGroupStyle}>
      <label style={labelStyle}>Phone</label>
      <input
        style={inputStyle}
        value={formData.phone}
        onChange={(e) => handleInputChange('phone', e.target.value)}
        placeholder="Enter phone number"
      />
    </div>
    <div style={fieldGroupStyle}>
      <label style={labelStyle}>Email</label>
      <input
        style={inputStyle}
        type="email"
        value={formData.email}
        onChange={(e) => handleInputChange('email', e.target.value)}
        placeholder="Enter email address"
      />
    </div>
  </div>
)
</div>

/* Comments Section */
<div style={sectionStyle}>
  <div
    style={sectionHeaderStyle}
    onClick={() => toggleSection('comments')}
  >
    <span>{expandIcon(isExpanded.comments)}</span>
    <span>Comments</span>
  </div>
  {isExpanded.comments && (
    <div style={{ padding: '12px' }}>
      <div style={fieldGroupStyle}>
        <label style={labelStyle}>Comments</label>
        <textarea
          style={textAreaStyle}
          value={formData.comments}
          onChange={(e) => handleInputChange('comments', e.target.value)}
          placeholder="Enter any additional comments or notes..."
        >
      </div>
    </div>
  )
}
```

```
        rows={4}
      />
    </div>
  </div>
})
</div>

{/* Action Buttons */}
<div style={{
  padding: '16px',
  backgroundColor: 'white',
  borderTop: '1px solid #d4d4d4',
  display: 'flex',
  gap: '8px',
  justifyContent: 'flex-end'
}}>
  <button style={{
    padding: '6px 16px',
    fontSize: '12px',
    border: '1px solid #ccc',
    backgroundColor: 'white',
    cursor: 'pointer'
}}>
    Cancel
  </button>
  <button style={{
    padding: '6px 16px',
    fontSize: '12px',
    border: '1px solid #4a6741',
    backgroundColor: '#4a6741',
    color: 'white',
    cursor: 'pointer'
}}>
    Save
  </button>
</div>

{/* Debug Info */}
<div style={{
  margin: '16px',
  padding: '12px',
  backgroundColor: '#f8f9fa',
  border: '1px solid #e9ecef',
  borderRadius: '4px',
  fontSize: '11px',
  color: '#6c757d'
}}>
  <strong>Form Data Preview:</strong>
  <pre style={{ marginTop: '8px', fontSize: '10px' }}>
    {JSON.stringify(formData, null, 2)}
  </pre>
</div>
</div>
);
```

```
};  
  
export default EditPageForm;
```