*Owen Lindsey*

*Professor Sparks, James*

*CST-391 Activity 5*

*11/16/2024*

**Part 1: Stopping Point #1 – Custom Components (Page 12)**   
  
**Creating a custom card component allows us to reuse code simply over our react application.**   
  
  
  
*Summary of new features that have been added.*

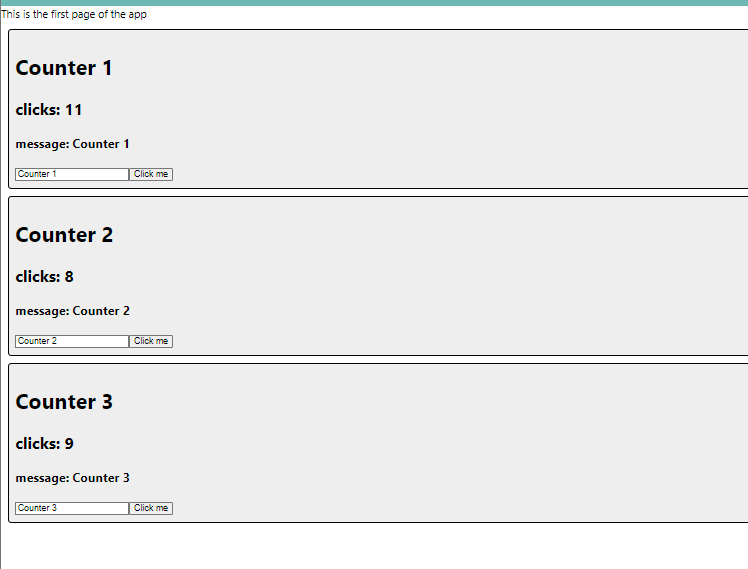
We created custom components, specifically a Card component, which demonstrates React's component-based architecture for reusable UI elements. We introduced the concept of 'props' (properties) to pass data from parent to child components, allowing us to populate each card with different album information including titles, descriptions, and images. The application uses JSX, React's syntax extension for JavaScript that allows us to write HTML-like code within JavaScript. We also integrated Bootstrap's CSS framework via CDN for styled components and responsive design.

*Definition of new technology used:*

'components' (reusable UI pieces)

'props' (data passed between components)

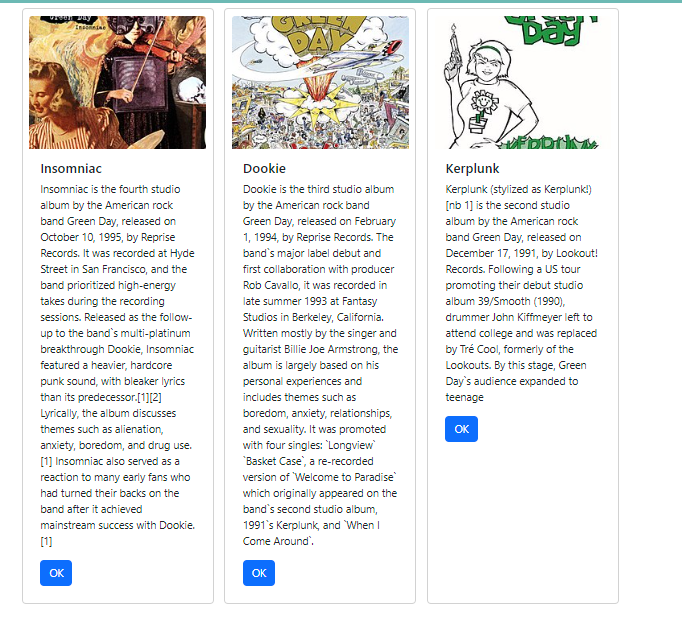
'JSX' (JavaScript XML syntax), and the concept of state management using the 'useState' hook.

**Part 1: State Changer Demo**  
  
  
  
  
  
  
*Summary of new features that have been added.*

We created a State Changer Demo that introduced React Hooks, specifically the useState Hook, which is a modern approach to handling state in functional components. The application demonstrated the key differences between 'props' (unchangeable data passed from parent to child components) and 'state' (mutable data managed within a component). We implemented two types of state management: a click counter and a message input, showcasing how useState can handle different types of data and user interactions.

*Definition of new technology used:*

**'Hooks'** (functions that allow functional components to use React state)  
  
**'useState'** (a specific Hook for state management), 'controlled components' (form elements whose values are controlled by React)  
  
**'event handlers'** (functions that respond to user actions). The application uses React's unidirectional data flow, where state changes propagate down through child components, causing the UI to update accordingly.

*Part 2: Stopping Point #2 – State and Props*  
  
  
  
*Summary of new features that have been added.*

we enhanced the music application by implementing state management and improving the visual layout. We introduced the concept of component state using the useState Hook, specifically creating an 'albumList' state variable to manage our album data. This demonstrated the key difference between props (static data passed from parent to child) and state (dynamic data managed within components). We explored the JavaScript map() function as a powerful tool for transforming data, using it to dynamically render multiple Card components from our album data. The application's visual presentation was improved through CSS Flexbox implementation, allowing cards to display horizontally instead of vertically.

*Definition of new technology used:*   
  
'state management' (handling dynamic data within components)

'map function' (array transformation method), 'Flexbox' (CSS layout model)

'container classes' (Bootstrap layout components). This allows for future improvements, including moving data to external sources and implementing REST service integration.