MovieLink must be a functional component. It will return markup similar to the following (though you will replace 1366 with the passed tmdbID value):

```
<a className="button card-footer-item"
   href="https://www.themoviedb.org/movie/1366" >
        <img src="images/tmdb.svg" width="30" />
</a>
```

3. In the App component, you will add the <MovieList> component to the render. Be sure to pass it the list of movies in state. Test.
4. In the App component, use map() to output a <MovieForm> for each movie. Be sure to pass both index and key values to each MovieForm. Also pass the saveChanges method to each MovieForm. Test.
5. Make MovieForm a Controlled Form Component. This will require creating some type of handler method within MovieForm that will call the saveChanges method that has been passed in (see also next step).
6. Implement saveChanges in the App component. Notice that it expects a movie object that contains within it the new data. Your method will use the index to replace the movie object from the movies data with the new data, and then update the state. Test.

Guidance and Testing

1. Break this problem down into smaller steps. Verify each component works as you create them.

PROJECT 2: Favorites List

DIFFICULTY LEVEL: Intermediate

Overview

This project builds on the completed Test Your Knowledge #2 by adding a favorites list.

Instructions

1. Use your completed Test Your Knowledge #2 as the starting point for this project.
2. The PhotoThumb component already has an Add Favorite button (the heart icon). You are going to implement its functionality. When the user clicks on this button, it will add a new favorite to an array stored in state, which should update the display in the Favorites component (it will be blank row at first between the header and the photo browser).
3. You will need to create two new components: one called FavoriteBar, the other called FavoriteItem (there is already a CSS class defined in the provided CSS named favorites). FavoriteBar should display a list of FavoriteItem components. Since the favorites CSS class uses grid layout, your FavoriteItem component will need to wrap the image in a block-level item, such as a <div>.

You will also need to add the `FavoriteBar` component to the `render` function of `App`.

4. For each thumbnail in `FavoriteItem`, you will need to add a click handler that will remove the image from the favorites list. The supplied CSS uses the :hover pseudo-element to visually indicate what will happen when the user clicks an image in the list.
5. Implement the hide/show functionality in `FavoriteBar`.

Guidance and Testing

1. Break this problem down into smaller steps. First verify the add to favorites and remove from favorites functions work simply by outputting the revised list to the console.
2. Once you are sure the add and remove functions work, implement the functionality in the new components.

PROJECT 3: Stock Dashboard

DIFFICULTY LEVEL: Advanced

Overview

In this project, you will be creating a more complex dashboard application. Figure 11.21 illustrates a wireframe diagram for this application. You will not be supplied any user interface: you can create your own styling or make use of a third-party React UI component library. The provided readme file contains URLs for the three APIs (Client API, Portfolio API, History API) and the location for the company logo images.

Instructions

1. This project is a single-page application containing three views: Home, Client, and Company. Your application must display different views depending on the left-side menu selection. It should default to the Home view.
2. When your application starts, it should display the Home view. This will contain just a single `<select>` list with a list of client names obtained from the client API. When the user selects a client, the Client view will be displayed.
3. The client view will display the information obtained from the Client API. The portfolio for the client can be obtained from the Portfolio API using the id for the current client. The entire client's stock portfolio details should be displayed in a table: it should display the company stock symbol, the company name, the current value of that stock from the History API, the number of that stocks in the portfolio, and the total value (number owned x current stock value) for this portfolio item. Display the top three stocks in terms of their total value near the top of the view (in the box, display the stock symbol and its total value in the portfolio. The portfolio summary should display the following values: total current portfolio value, the number of portfolio items, total number of stocks.