Practice Hands-on lab: Budget Allocation Application

Estimated Time: 60 minutes

In this React Budget Allocation app, you will learn how to break down a UI into React components. You will become familiar with state using the Context API. Furthermore, you will explore actions, reducers, and the dispatch function.You will create a code file, save it, and edit it to make changes.

Objectives:

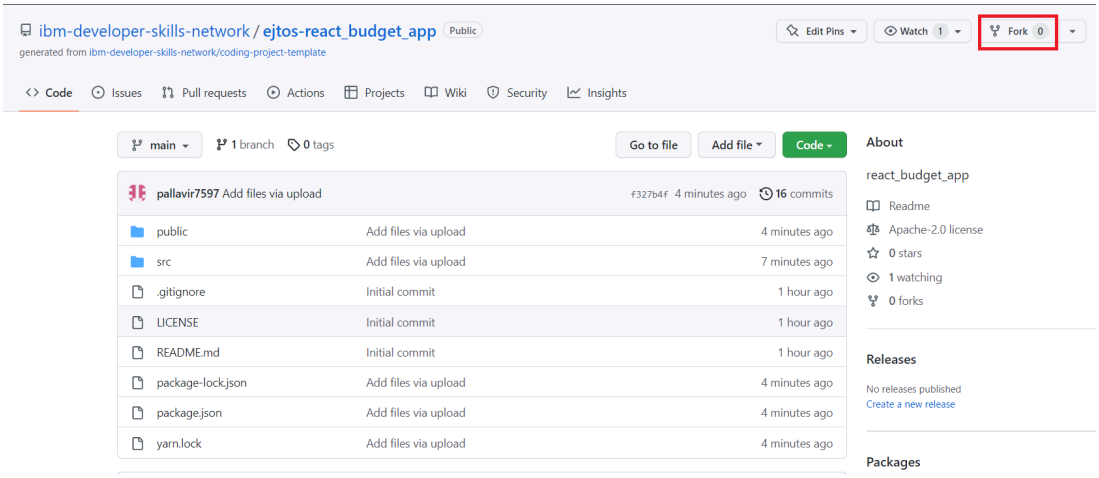
- Setup a React Project.
- Put the UI Components in Place.
- Render the created components and context in App.js.
- View your app on the browser.

Exercise 1: Setup a React Project

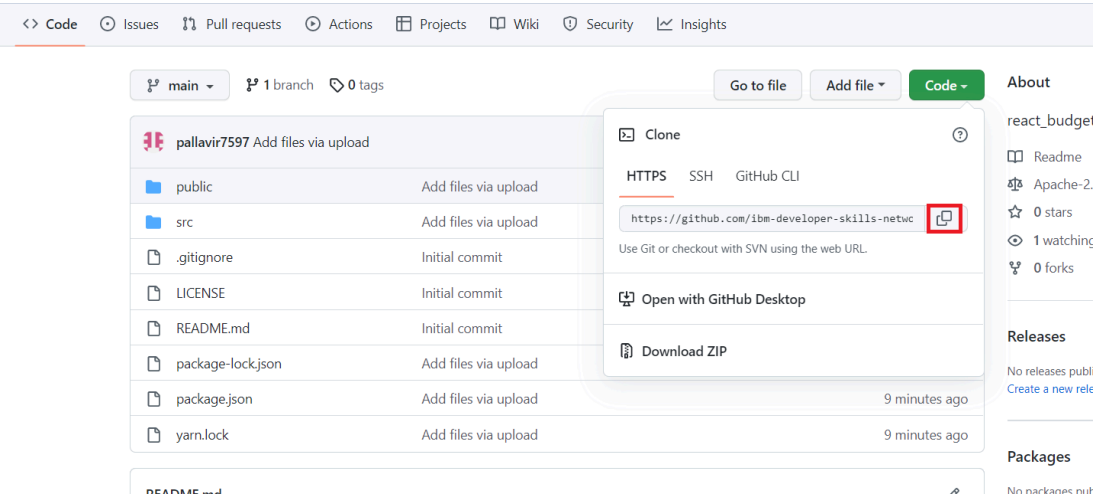
Fork the Git repository to have the react code you need to start

1. Go to the project repository on this link which has the partially developed code for react code.

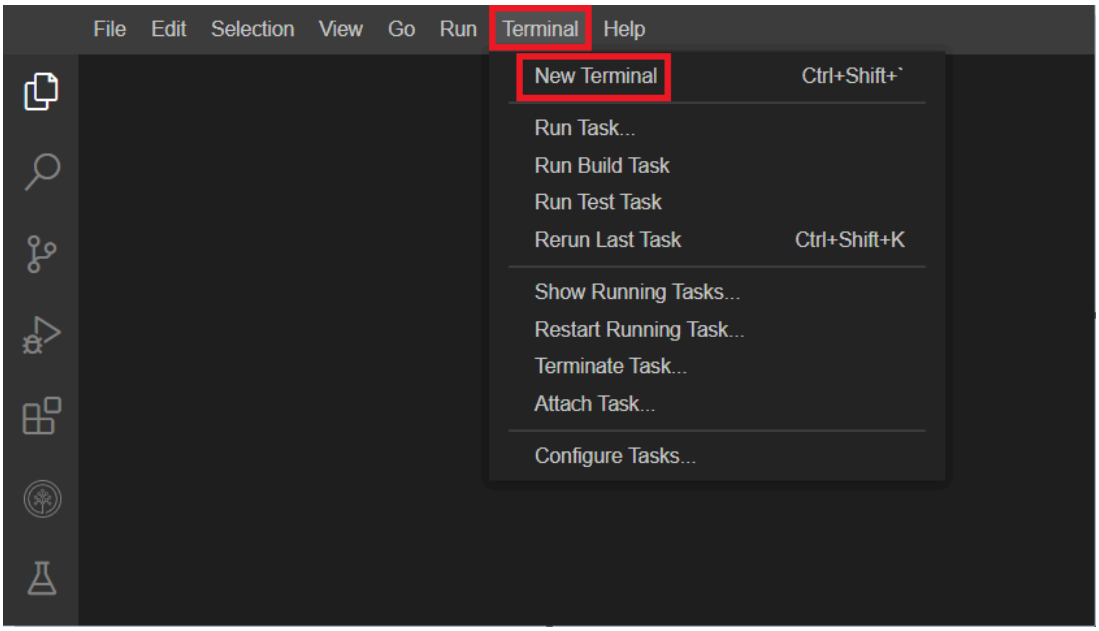
2. Create a fork of the repository into your own. *You will need to have a github account of your own to do so.*



3. Go to your repository and copy the clone url.



4. Open a terminal window by using the menu in the editor: Terminal > New Terminal.

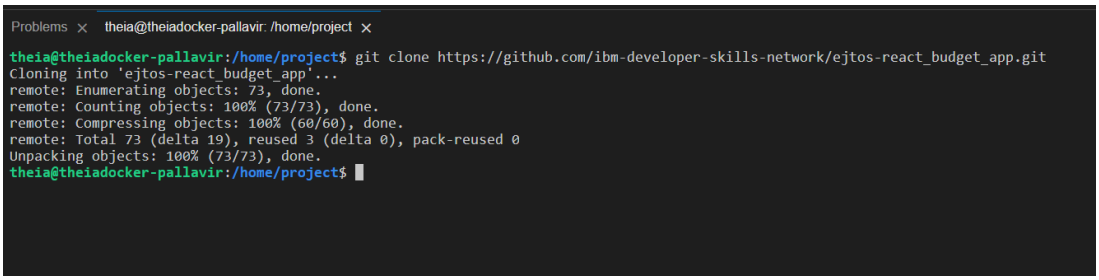


5. Clone the repository by running the command given below:

```
1. 1
```

```
1. git clone <your_repo_name>
```

Copied!



6. This will clone the repository with budget allocation app files in your home directory in the lab environment. Check the same with the following command.

```
1. 1
```

```
1. ls
```

Copied! Executed!

7. Change to the project directory and check its contents.

```
1. 1
```

```
1. cd ejtos-react_budget_app && ls
```

Copied! Executed!

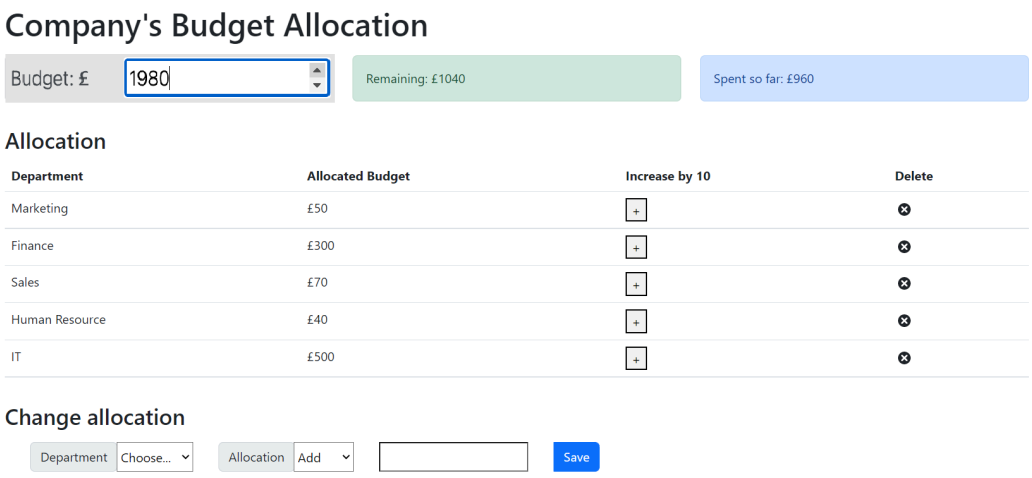
8. All packages required to be installed are listed in package.json. Run npm install -s, to install and save those packages.

```
1. 1
```

```
1. npm install  -s
```

Copied! Executed!

Exercise 2: Put the UI Components in Place.



The image above displays the Company's Budget Allocation application page you will be developing in this practice project.

It has the following six components:

- Budget
- ExpenseItem
- ExpenseList
- ExpenseTotal (Spent so far)
- Remaining
- AllocationForm

All of these components will be using redux for state management through AppContext.js. You can verify the content of AppContext.js by clicking on the below button.

- In `AppContext.js` you will be creating reducer, which is used to update the state, based on the action. Then you will set the initial state for the departments. You will be creating the Provider component which wraps the components you want to give access to the state.
- Here, you are adding an initial budget, creating a Provider component, setting up the useReducer hook which will hold your state, and allow you to update the state via dispatch.
- Adding a new case to the switch statement called "ADD_EXPENSE", "RED_EXPENSE", "DELETE_EXPENSE".

Open **AppContext.js** in IDE

Details of the six components:

- In `Budget.js` you will be adding text and value for your budget. You will be importing app context and the useContext hook, and pass your AppContext to it - this is how a component connects to the context in order to get values from global state.
- In `Remaining.js` you will be importing expense and budget from context and getting the remaining value using subtraction.
- In `ExpenseTotal.js` you will be adding useContext and AppContext. Taking the expenses from state.
- In `ExpenseList.js` you will be using the map function to display the Expenseitem component.
- In `Expenseitem.js` you will be importing dispatch from Context, which allows you to dispatch a delete action, creating a function that gets called when the delete icon is clicked.
- In `AllocationForm.js` you will be creating an expense object, containing the name and the cost. This is what will get dispatched as the payload, and what you'll use to update the state.

At start you will be creating a `Budget`, `Remaining` and `Spent so far` button.

1. Make changes to **Budget.js** component. Open the code, in the `src/components/Budget.js`. Copy the below code and paste in the Budget.js

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
```

```
13
14
15
16
17
```

```
import React, { useContext, useState } from 'react';
import { AppContext } from '../context/AppContext';

const Budget = () => {
    const { budget } = useContext(AppContext);
    const [newBudget, setNewBudget] = useState(budget);
    const handleBudgetChange = (event) => {
        setNewBudget(event.target.value);
    }
    return (
<div className='alert alert-secondary'>
<span>Budget: £{budget}</span>
<input type="number" step="10" value={newBudget} onChange={handleBudgetChange}></input>
</div>
    );
};
export default Budget;
```

Copied!

In the above code snippet we are using the useState hook to create a state variable called newBudget and initialize it with the current value of budget. We are also defining a function called handleBudgetChange that updates the value of newBudget when the user changes the value of the input field.

We are then setting the value attribute of the input field to newBudget and adding an onChange event listener that calls handleBudgetChange when the user changes the value of the input field.

Here, you are using the Bootstrap Alert classes to give a nice gray background by adding some text and hard coding a value.

2. Make changes to App.js. Open the code, in the `src/App.js`. Copy the below code and paste in the App.js in the space provided. You will notice that `Budget.js` has been already imported for you in App.js.

```
1
2
3
4
```

```
                // Budget component
                    <div className='col-sm'>
                        <Budget />
                    </div>
```

Copied!

3. Make sure you are in the `ejtos-react_budget_app` directory and run the server using the following command.

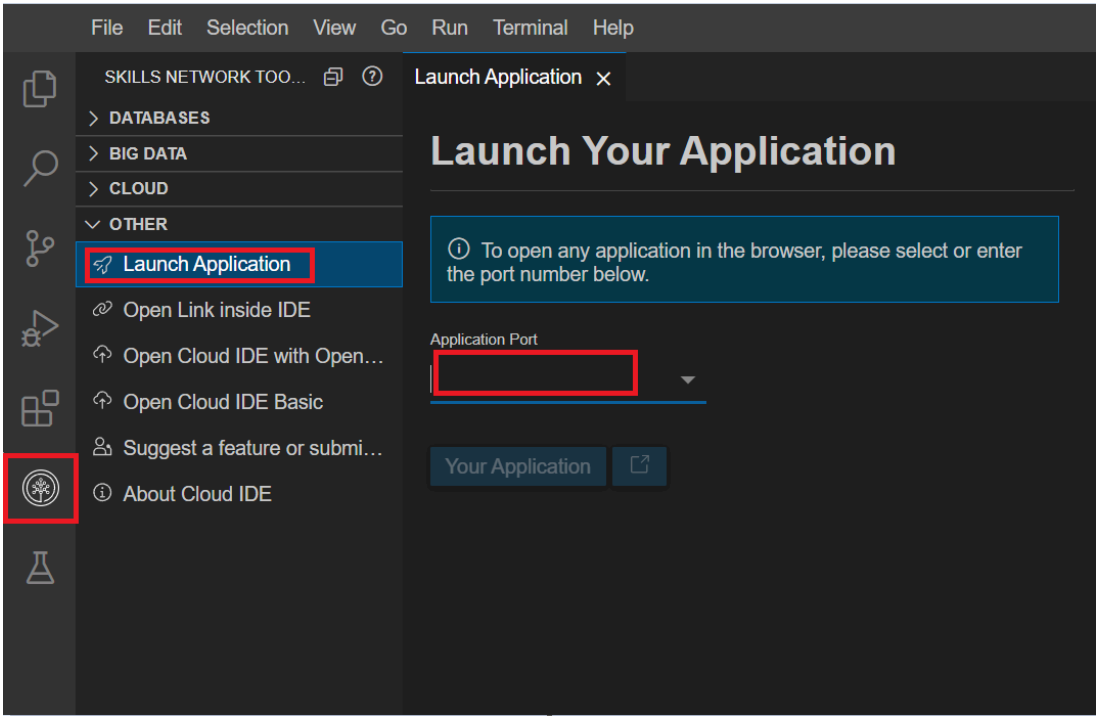
```
1
```

```
npm start
```

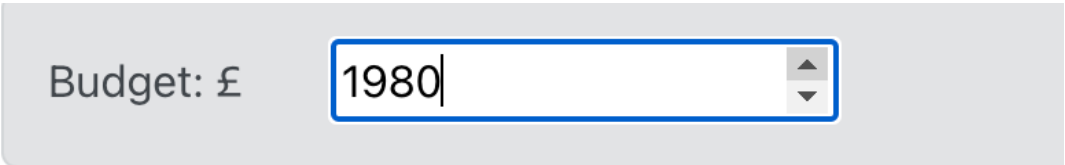
Copied! Executed!

4. Click on the button below to launch the application or click on Skills Network button on the left, it will open the **"Skills Network Toolbox"**. Then click **Other** then **Launch Application**. From there you should be able to enter the port `3000`.

Budget Application



5. Verify your output on the browser.



You can view the Budget button created in the above image.

6. Make changes to **Remaining.js** component. Open the code, in the `src/components/Remaining.js`. Copy the below code and paste in the Remaining.js

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

```
import React, { useContext } from 'react';
import { AppContext } from '../context/AppContext';
const Remaining = () => {
    const { expenses, budget } = useContext(AppContext);
    const totalExpenses = expenses.reduce((total, item) => {
        return (total = total + item.cost);
    }, 0);
    const alertType = totalExpenses > budget ? 'alert-danger' : 'alert-success';
    return (
        <div className={`alert ${alertType}`}>
            <span>Remaining: £{budget - totalExpenses}</span>
        </div>
    );
};
export default Remaining;
```

Copied!

Here, you are using the reduce function to get a total of all the costs, assigning this to a variable and displaying the variable in your JSX.
Now whenever the user adds an expense, this causes the state to update, which will cause all components connected to the context to re-render and update themselves with new values.

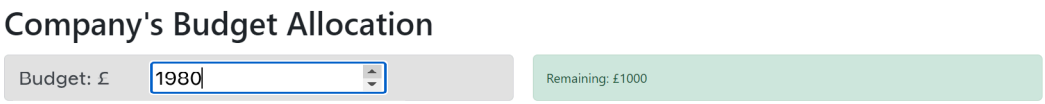
7. Make changes to App.js. Open the code, in the `src/App.js`. Copy the below code and paste in the App.js in the space provided. Import the component in the appropriate place using `import Remaining from './components/Remaining';`

```
1
2
3
4
```

```
                //Remaining component
                <div className='col-sm'>
                    <Remaining />
                </div>
```

Copied!

8. Refresh the React app page launched in step 4 and verify the output.



9. Make changes to **ExpenseTotal.js** component. Open the code, in the `src/components/ExpenseTotal.js`. Copy the below code and paste in the ExpenseTotal.js

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

```
import React, { useContext } from 'react';
import { AppContext } from '../context/AppContext';
const ExpenseTotal = () => {
    const { expenses } = useContext(AppContext);
    const totalExpenses = expenses.reduce((total, item) => {
        return (total += item.cost);
    }, 0);
    return (
        <div className='alert alert-primary'>
            <span>Spent so far: £{totalExpenses}</span>
        </div>
    );
};
export default ExpenseTotal;
```

Copied!

Here, you are using the reduce function to get a total of all the costs, assigning this to a variable and displaying the variable in your JSX.
Now whenever the user adds an expense, this causes the state to update, which will cause all components connected to the context to re-render and update themselves with new values.

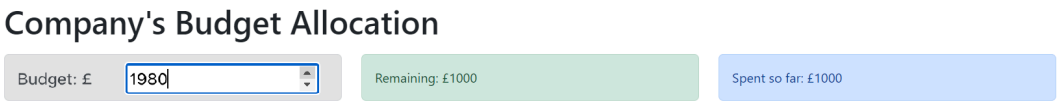
10. Make changes to App.js. Open the code, in the `src/App.js`. Copy the below code and paste in the App.js in the space provided.Import the component in the appropriate place using `import ExpenseTotal from './components/ExpenseTotal';`

```
1
2
3
4
```

```
                //ExpenseTotal component
                   <div className='col-sm'>
                       <ExpenseTotal />
                   </div>
```

Copied!

11. Refresh the React app page launched in step 4 and verify the output.



12. Make changes to **ExpenseList.js** component. Open the code, in the `src/components/ExpenseList.js`. Copy the below code and paste in the ExpenseList.js.

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
```

```
import React, { useContext } from 'react';
import ExpenseItem from './ExpenseItem';
import { AppContext } from '../context/AppContext';

const ExpenseList = () => {
    const { expenses } = useContext(AppContext);

    return (
        <table className='table'>
              <thead className="thead-light">
            <tr>
              <th scope="col">Department</th>
              <th scope="col">Allocated Budget</th>
              <th scope="col">Increase by 10</th>
              <th scope="col">Delete</th>
```

```
            </tr>
          </thead>
            <tbody>
            {expenses.map((expense) => (
                <ExpenseItem id={expense.id} key={expense.id} name={expense.name} cost={expense.cost} />
            ))}
            </tbody>
        </table>
    );
};

export default ExpenseList;
```

Copied!

Here, you are creating a list, using the map function to iterate over the expenses, and displaying an ExpenseItem component.

13. Make changes to **ExpenseItem.js** component. Open the code, in the `src/components/ExpenseItem.js`. Copy the below code and paste in the ExpenseItem.js.

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
```

```
import React, { useContext } from 'react';
import { TiDelete } from 'react-icons/ti';
import { AppContext } from '../context/AppContext';

const ExpenseItem = (props) => {
    const { dispatch } = useContext(AppContext);

    const handleDeleteExpense = () => {
        dispatch({
            type: 'DELETE_EXPENSE',
            payload: props.id,
        });
    };

    const increaseAllocation = (name) => {
        const expense = {
            name: name,
            cost: 10,
        };

        dispatch({
            type: 'ADD_EXPENSE',
            payload: expense
        });

```

```
    }

    return (
        <tr>
        <td>{props.name}</td>
        <td>£{props.cost}</td>
        <td><button onClick={event=> increaseAllocation(props.name)}>+</button></td>
        <td><TiDelete size='1.5em' onClick={handleDeleteExpense}></TiDelete></td>
        </tr>
    );
};

export default ExpenseItem;
```

Copied!

Here, you are dispatching an action. Your action contains the type (so the reducer knows how to update the state) and the payload. In this case you are passing the ID of this expense (which you get from props when you rendered the ExpenseList).

14. Make changes to **AllocationForm.js** component. Open the code, in the `src/components/AllocationForm.js`. Copy the below code and paste in the AllocationForm.js.

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
```

```
62. 62
63. 63
64. 64
65. 65
66. 66
67. 67
68. 68
69. 69
70. 70
71. 71
72. 72
73. 73
74. 74
75. 75
76. 76
77. 77
78. 78
79. 79
80. 80
81. 81
```

```
 1. import React, { useContext, useState } from 'react';
 2. import { AppContext } from '../context/AppContext';
 3.
 4. const AllocationForm = (props) => {
 5.     const { dispatch,remaining  } = useContext(AppContext);
 6.
 7.     const [name, setName] = useState('');
 8.     const [cost, setCost] = useState('');
 9.     const [action, setAction] = useState('');
10.
11.     const submitEvent = () => {
12.
13.             if(cost > remaining) {
14.                 alert("The value cannot exceed remaining funds  £"+remaining);
15.                 setCost("");
16.                 return;
17.             }
18.
19.         const expense = {
20.             name: name,
21.             cost: parseInt(cost),
22.         };
23.         if(action === "Reduce") {
24.             dispatch({
25.                 type: 'RED_EXPENSE',
26.                 payload: expense,
27.             });
28.         } else {
29.                 dispatch({
30.                     type: 'ADD_EXPENSE',
31.                     payload: expense,
32.                 });
33.             }
34.     };
35.
36.     return (
37.         <div>
38.             <div className='row'>
39.
40.             <div className="input-group mb-3" style={{ marginLeft: '2rem' }}>
41.                     <div className="input-group-prepend">
42.                 <label className="input-group-text" htmlFor="inputGroupSelect01">Department</label>
43.                   </div>
44.                   <select className="custom-select" id="inputGroupSelect01" onChange={(event) => setName(event.target.value)}>
45.                         <option defaultValue>Choose...</option>
46.                         <option value="Marketing" name="marketing"> Marketing</option>
47.                 <option value="Sales" name="sales">Sales</option>
48.                 <option value="Finance" name="finance">Finance</option>
49.                 <option value="HR" name="hr">HR</option>
50.                 <option value="IT" name="it">IT</option>
51.                 <option value="Admin" name="admin">Admin</option>
52.                   </select>
53.
54.                     <div className="input-group-prepend" style={{ marginLeft: '2rem' }}>
55.                 <label className="input-group-text" htmlFor="inputGroupSelect02">Allocation</label>
56.                   </div>
57.                   <select className="custom-select" id="inputGroupSelect02" onChange={(event) => setAction(event.target.value)}>
58.                         <option defaultValue value="Add" name="Add">Add</option>
59.                 <option value="Reduce" name="Reduce">Reduce</option>
60.                   </select>
61.
62.                     <input
63.                         required='required'
64.                         type='number'
```

```
                        id='cost'
                        value={cost}
                        style={{ marginLeft: '2rem' , size: 10}}
                        onChange={(event) => setCost(event.target.value)}>
                        </input>

                    <button className="btn btn-primary" onClick={submitEvent} style={{ marginLeft: '2rem' }}>
                        Save
                    </button>
                </div>
                </div>

        </div>
    );
};
export default AllocationForm;
```

Copied!

Here, you are adding form tags, adding a label/input for name, cost and action field, and adding values for various departments.

Excercise 3: Render the created components and context in App.js

1. Make changes to **App.js**. Open the code, in the `src/App.js`. You need to add the code in the container div to import and add the components created above.

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
```

```
import React from 'react';

import 'bootstrap/dist/css/bootstrap.min.css';

//Code to import Budget.js
import Budget from './components/Budget';

// Add code to import the other components here under


import { AppProvider } from './context/AppContext';
const App = () => {
    return (
        <AppProvider>
            <div className='container'>
                <h1 className='mt-3'>Company's Budget Allocation</h1>
                    <div className='row mt-3'>
                        {/* Add Budget component here under */}

```

```
                        {/* Add Remaining component here under */}

                        {/* Add ExpenseTotal component here under */}

                        {/* Add ExpenseList component here under */}

                        {/* Add ExpenseItem component here under */}

                        {/* Add AllocationForm component here under */}

                </div>
            </div>
        </AppProvider>
    );
};
export default App;
```

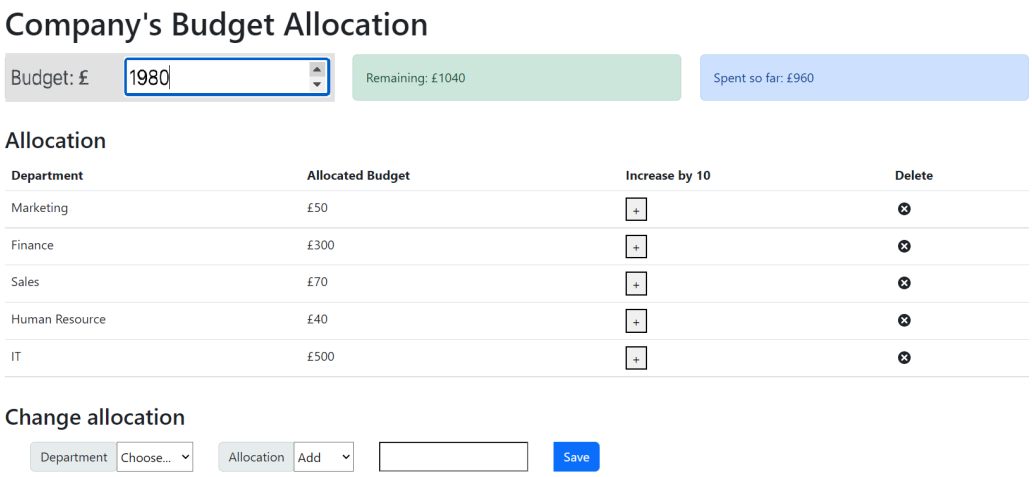
Copied!

▶ Click here to view the hint
▶ Click here to view the solution

Here, you are importing different components,adding a bootstrap container that helps you center your App on the page

- Adding a title
- Adding a Bootstrap row
- Adding a column within the row for each of your components so far
- Imported and Rendered the AllocationForm
- Imported AppProvider and Nested components in the AppProvider element.

2. Refresh the React app page launched and verify the output.



Commit and push your local code to your remote git repository

This git pushed code will be cloned and used in final project.

Please refer to this lab Working with git in the Theia lab environment

Congratulations! You have completed this practice lab and know how to create components of a budget allocation application.

Author(s)

Pallavi Rai

Change Log

Date	Version	Changed by	Change Description
2022-08-31	1.0	Pallavi Rai	Initial version created
2023-09-05	1.1	Sapthashree & Ritika	Updated Budget.js code