```
// Dynamic tabs
_____________
import React, { useState } from 'react';
import PropTypes from 'prop-types';
import { AppBar, Tabs, Tab, Grid, Box, Typography } from '@material-ui/core';
import Close from '@material-ui/icons/Close';

function TabPanel(props) {
  const { children, value, index, ...other } = props;

  return (
    <div
      role="tabpanel"
      hidden={value !== index}
      id={`scrollable-auto-tabpanel-${index}`}
      aria-labelledby={`scrollable-auto-tab-${index}`}
      {...other}
    >
      {value === index && (
        <Box p={3}>
          <Typography>{children}</Typography>
        </Box>
      )}
    </div>
  );
}

TabPanel.propTypes = {
  children: PropTypes.node,
  index: PropTypes.any.isRequired,
  value: PropTypes.any.isRequired,
};

const menuItems = ['Dashboard', 'Applications', 'Incoming', 'Services'];

const DashboardSupply = () => {
  const [tabList, setTabList] = useState([
    {
      key: 0,
      id: 0,
      label: 'Dashboard',
    },
  ]);
```

```
const [tabValue, setTabValue] = useState(0);
const [value, setValue] = React.useState(0);

const handleTabChange = (event, value) => {
  setTabValue(value);
  let labels = Object.keys(tabList).map((val) => tabList[val].label);
  let label = labels[value];
  let index = menuItems.indexOf(label);
  setValue(index);
};

const handleChange = (event, newValue) => {
  setValue(newValue);
  let labels = Object.keys(tabList).map((val) => tabList[val].label);

  let tabId = labels.indexOf(menuItems[newValue]);
  if (tabId !== -1) {
    setTabValue(tabId);
  } else {
    addTab(newValue);
  }
};

const addTab = (value) => {
  let id = tabList[tabList.length - 1].id + 1;
  let labels = Object.keys(tabList).map((val) => tabList[val].label);
  let tabId = labels.indexOf(menuItems[value]);

  if (
    labels.indexOf(menuItems[value]) === -1 &&
    labels.length < menuItems.length
  ) {
    setTabList([...tabList, { key: id, id: id, label: menuItems[value] }]);
    setTabValue(tabList.length);
  } else {
    setTabValue(labels.indexOf(menuItems[value]));
  }
};

const deleteTab = (e) => {
  e.stopPropagation();
  let labels = Object.keys(tabList).map((val) => tabList[val].label);

  if (tabList.length === 1) {
```

```
    return;
  }
  let tabId = parseInt(e.currentTarget.id);
  let tabIDIndex = 0;

  let tabs = tabList.filter((value, index) => {
    if (value.id === tabId) {
      tabIDIndex = index;
    }
    return value.id !== tabId;
  });

  console.log('--- tabId  ---', tabId);
  let index = menuItems.indexOf(tabs[tabId - 1].label);

  tabs.map((item, index) => {
    item.key = index;
    item.id = index;
  });
  let currentValue = parseInt(tabValue);
  if (currentValue === tabId) {
    if (tabIDIndex === 0) {
      currentValue = tabList[tabIDIndex + 1].id;
    } else {
      currentValue = tabList[tabIDIndex - 1].id;
    }
  } else {
    if (currentValue > tabId) {
      currentValue = parseInt(tabValue) - 1;
      index = menuItems.indexOf(tabs[currentValue].label);
    }
  }
  setTabValue(currentValue);
  setValue(index);
  setTabList(tabs);
};

return (
  <>
    <AppBar position="static">
      <Grid container alignItems="center" justify="center">
        <Grid item xl={11} lg={11} md={11} sm={11} xs={11}>
          <Tabs
            value={tabValue}
```

```
        onChange={handleTabChange}
        variant="scrollable"
        scrollButtons="auto"
        style={{ paddingLeft: 176 }}
      >
        {tabList.map((tab) => (
          <Tab
            key={tab.key.toString()}
            value={tab.id}
            label={tab.label}
            icon={tab.id > 0 && <Close id={tab.id} onClick={deleteTab} />}
            className="mytab"
            style={{ color: 'yellow' }}
          />
        ))}
      </Tabs>
    </Grid>
    <Grid item xl={11} lg={11} md={11} sm={11} xs={11}>
      <Tabs
        orientation="vertical"
        variant="scrollable"
        value={value}
        onChange={handleChange}
        aria-label="Vertical tabs example"
      >
        {menuItems.map((tab, i) => (
          <Tab
            key={i}
            value={i}
            label={menuItems[i]}
            className="mytab"
            style={{ color: 'yellow' }}
          />
        ))}
      </Tabs>
    </Grid>
  </Grid>
</AppBar>
<TabPanel value={value} index={0}>
  Item One
</TabPanel>
<TabPanel value={value} index={1}>
  Item Two
</TabPanel>
```

```
      <TabPanel value={value} index={2}>
        Item Three
      </TabPanel>
      <TabPanel value={value} index={3}>
        Item Four
      </TabPanel>
      <TabPanel value={value} index={4}>
        Item Five
      </TabPanel>
      <TabPanel value={value} index={5}>
        Item Six
      </TabPanel>
      <TabPanel value={value} index={6}>
        Item Seven
      </TabPanel>
    </>
  );
};

export default DashboardSupply;
```

```
import React, { useState } from 'react';
import { AppBar, Tabs, Tab, Grid, Button } from '@material-ui/core';
import Add from '@material-ui/icons/Add';
import Close from '@material-ui/icons/Close';

const menuItems = ['Dashboard', 'Applications', 'Incoming', 'Services'];

const DashboardSupply = () => {
  const [tabList, setTabList] = useState([
    {
      key: 0,
      id: 0,
      label: 'Dashboard',
    },
  ]);

  const [tabValue, setTabValue] = useState(0);
  const [value, setValue] = React.useState(0);

  const handleTabChange = (event, value) => {
    setTabValue(value);
    let labels = Object.keys(tabList).map((val) => tabList[val].label);
    let label = labels[value];
```

```
  let index = menuItems.indexOf(label);
  setValue(index);
  console.log('value tab change', value);
};

const handleChange = (event, newValue) => {
  setValue(newValue);
  let labels = Object.keys(tabList).map((val) => tabList[val].label);

  let tabId = labels.indexOf(menuItems[newValue]);
  if (tabId !== -1) {
    setTabValue(tabId);
  } else {
    addTab(newValue);
  }
};

const addTab = (value) => {
  let id = tabList[tabList.length - 1].id + 1;
  let labels = Object.keys(tabList).map((val) => tabList[val].label);
  let tabId = labels.indexOf(menuItems[value]);

  if (
    labels.indexOf(menuItems[value]) === -1 &&
    labels.length < menuItems.length
  ) {
    setTabList([...tabList, { key: id, id: id, label: menuItems[value] }]);
    setTabValue(tabList.length);
  } else {
    setTabValue(labels.indexOf(menuItems[value]));
  }
};

const deleteTab = (e) => {
  e.stopPropagation();
  let labels = Object.keys(tabList).map((val) => tabList[val].label);

  if (tabList.length === 1) {
    return;
  }
  //let tabId = parseInt(e.target.id);
  let tabId = parseInt(e.currentTarget.id);
  console.log('tabId delete init:', tabId);
  let tabIDIndex = 0;
```

```
let tabs = tabList.filter((value, index) => {
  if (value.id === tabId) {
    tabIDIndex = index;
  }
  return value.id !== tabId;
});

console.log('--- tabId  ---', tabId);
let index = menuItems.indexOf(tabs[tabId - 1].label);

tabs.map((item, index) => {
  item.key = index;
  item.id = index;
});
let currentValue = parseInt(tabValue);
if (currentValue === tabId) {
  if (tabIDIndex === 0) {
    console.log('tabId delete if:', tabId);
    currentValue = tabList[tabIDIndex + 1].id;
  } else {
    console.log('tabId delete else:', tabId);
    currentValue = tabList[tabIDIndex - 1].id;
  }
} else {
  if (currentValue > tabId) {
    console.log('tabId delete last else:', tabId);
    console.log('tabs delete last else:', tabs);
    console.log('currentValue delete last else:', currentValue);
    //currentValue = tabId;
    currentValue = parseInt(tabValue) - 1;
    index = menuItems.indexOf(tabs[currentValue].label);
  }
}
console.log('currentValue outside delete:', currentValue);
//console.log('label delete:', e.currentTarget.name);
//setTabValue(labels.indexOf(menuItems[e.currentTarget.id]));
//console.log('index delete:', labels.indexOf(e.currentTarget.label));
//let index = menuItems.indexOf(e.currentTarget.label);

setTabValue(currentValue);
//setTabValue(tabId - 1);

setValue(index);
```

```
  setTabList(tabs);
};

return (
  <AppBar position="static">
    <Grid container alignItems="center" justify="center">
      <Grid item xl={11} lg={11} md={11} sm={11} xs={11}>
        <Tabs
          value={tabValue}
          onChange={handleTabChange}
          variant="scrollable"
          scrollButtons="auto"
          style={{ paddingLeft: 176 }}
        >
          {tabList.map((tab) => (
            <Tab
              key={tab.key.toString()}
              value={tab.id}
              label={tab.label}
              icon={tab.id > 0 && <Close id={tab.id} onClick={deleteTab} />}
              className="mytab"
              style={{ color: 'yellow' }}
            />
          ))}
        </Tabs>
      </Grid>
      <Grid item xl={11} lg={11} md={11} sm={11} xs={11}>
        <Tabs
          orientation="vertical"
          variant="scrollable"
          value={value}
          onChange={handleChange}
          aria-label="Vertical tabs example"
        >
          {menuItems.map((tab, i) => (
            <Tab
              key={i}
              value={i}
              label={menuItems[i]}
              className="mytab"
              style={{ color: 'yellow' }}
            />
          ))}
        </Tabs>
```

```
        </Grid>
        <Grid item xl={1} lg={1} md={1} sm={1} xs={1}>
          {/*<Button variant="outlined" onClick={addTab}>
            <Add />
            </Button>*/}
        </Grid>
      </Grid>
    </AppBar>
  );
};

export default DashboardSupply;
```

```
import React, { useState } from 'react';
import { AppBar, Tabs, Tab, Grid, Button } from '@material-ui/core';
import Add from '@material-ui/icons/Add';
import Close from '@material-ui/icons/Close';

const menuItems = ['Dashboard', 'Applications', 'Incoming', 'Services'];

const DashboardSupply = () => {
  const [tabList, setTabList] = useState([
    {
      key: 0,
      id: 0,
      label: 'Dashboard',
    },
  ]);

  const [tabValue, setTabValue] = useState(0);
  const [value, setValue] = React.useState(0);
  const handleTabChange = (event, value) => {
    setTabValue(value);
    let labels = Object.keys(tabList).map((val) => tabList[val].label);
    let label = labels[value];
    let index = menuItems.indexOf(label);
    console.log(index);
    setValue(index);
    //handleChange(value);
  };
  const handleChange = (event, newValue) => {
    setValue(newValue);
    console.log('newValue handleChange', newValue);
    let labels = Object.keys(tabList).map((val) => tabList[val].label);
```

```
  console.log('labels handleChange', labels);
  console.log('menuItems[newValue] handleChange', menuItems[newValue]);
  console.log(
    'indexof item in labels handleChange',
    labels.indexOf(menuItems[newValue]),
  );
  let tabId = labels.indexOf(menuItems[newValue]);
  if (tabId !== -1) {
    setTabValue(tabId);
    console.log('tabValue no add', tabValue);
    console.log('tabId no add', tabId);
  } else {
    console.log('else add');
    addTab(newValue);
  }
  console.log('tabValue change', tabValue);
};

const addTab = (value) => {
  console.log('tabValue before add:', tabValue);

  let id = tabList[tabList.length - 1].id + 1;
  let labels = Object.keys(tabList).map((val) => tabList[val].label);
  console.log('indexof item in labels add', labels.indexOf(menuItems[value]));
  let tabId = labels.indexOf(menuItems[value]);

  if (
    labels.indexOf(menuItems[value]) === -1 &&
    labels.length < menuItems.length
  ) {
    setTabList([...tabList, { key: id, id: id, label: menuItems[value] }]);
    console.log('value add', value, 'tabList.length add', tabList.length);
    setTabValue(tabList.length);
  } else {
    setTabValue(labels.indexOf(menuItems[value]));
  }

  console.log('add tabList length', tabList.length);
  //let id = tabList[tabList.length - 1].id + 1;
  //setTabList([...tabList, { key: id, id: id }]);
};

const deleteTab = (e) => {
  e.stopPropagation();
```

```
console.log('delete tabList length before delete:', tabList.length);
let labels = Object.keys(tabList).map((val) => tabList[val].label);

if (tabList.length === 1) {
  return;
}
//let tabId = parseInt(e.target.id);
let tabId = parseInt(e.currentTarget.id);
console.log('tabId delete:', tabId);
let tabIDIndex = 0;

let tabs = tabList.filter((value, index) => {
  if (value.id === tabId) {
    tabIDIndex = index;
  }
  return value.id !== tabId;
});

console.log('---', tabs[tabId - 1].label);
let index = menuItems.indexOf(tabs[tabId - 1].label);

tabs.map((item, index) => {
  item.key = index;
  item.id = index;
});
let currentValue = parseInt(tabValue);
console.log('tabs before if delete currentValue:', tabs, currentValue);
/*if (currentValue === tabId) {
  if (tabIDIndex === 0) {
    currentValue = tabList[tabIDIndex + 1].id;
  } else {
    currentValue = tabList[tabIDIndex - 1].id;
  }
}*/
console.log('tabs delete:', tabs);
console.log('currentValue delete:', currentValue);
//console.log('label delete:', e.currentTarget.name);
//setTabValue(labels.indexOf(menuItems[e.currentTarget.id]));
console.log('tabs after if delete currentValue:', tabs, currentValue);
//console.log('index delete:', labels.indexOf(e.currentTarget.label));
//let index = menuItems.indexOf(e.currentTarget.label);

//setTabValue(currentValue - 1);
setTabValue(tabId - 1);
```

```
  setValue(index);
  setTabList(tabs);
};

return (
  <AppBar position="static">
    <Grid container alignItems="center" justify="center">
      <Grid item xl={11} lg={11} md={11} sm={11} xs={11}>
        <Tabs
          value={tabValue}
          onChange={handleTabChange}
          variant="scrollable"
          scrollButtons="auto"
          style={{ paddingLeft: 176 }}
        >
          {tabList.map((tab) => (
            <Tab
              key={tab.key.toString()}
              value={tab.id}
              label={tab.label}
              icon={tab.id > 0 && <Close id={tab.id} onClick={deleteTab} />}
              className="mytab"
              style={{ color: 'yellow' }}
            />
          ))}
        </Tabs>
      </Grid>
      <Grid item xl={11} lg={11} md={11} sm={11} xs={11}>
        <Tabs
          orientation="vertical"
          variant="scrollable"
          value={value}
          onChange={handleChange}
          aria-label="Vertical tabs example"
        >
          {menuItems.map((tab, i) => (
            <Tab
              key={i}
              value={i}
              label={menuItems[i]}
              className="mytab"
              style={{ color: 'yellow' }}
            />
          ))}
```

```
          </Tabs>
        </Grid>
        <Grid item xl={1} lg={1} md={1} sm={1} xs={1}>
          {/*<Button variant="outlined" onClick={addTab}>
            <Add />
            </Button>*/}
        </Grid>
      </Grid>
    </AppBar>
  );
};

export default DashboardSupply;
```

```
if (tabId !== -1) {
      setTabValue(tabId);
      console.log('tabValue no add', tabValue);
      console.log('tabId no add', tabId);
    } else {
      console.log('else add');
      addTab(newValue);
    }
```

```
//setTabValue(tabId);

    console.log('newValue change:', newValue);
    console.log('newValue item change 2:', menuItems[newValue]);
    console.log(
      'indexof item in labels change',
      labels.indexOf(menuItems[newValue]),
    );
```

```
import React, { useState } from 'react';
import { AppBar, Tabs, Tab, Grid, Button } from '@material-ui/core';
import Add from '@material-ui/icons/Add';
import Close from '@material-ui/icons/Close';

const menuItems = ['Dashboard', 'Applications', 'Incoming', 'Services'];

const DashboardSupply = () => {
  const [tabList, setTabList] = useState([
    {
      key: 0,
      id: 0,
```

```
    label: 'Dashboard',
  },
]);

const [tabValue, setTabValue] = useState(0);
const [value, setValue] = React.useState(0);
const handleTabChange = (event, value) => {
  setTabValue(value);
  //setValue(value);
  //handleChange(value);
};
const handleChange = (event, newValue) => {
  setValue(newValue);
  let labels = Object.keys(tabList).map((val) => tabList[val].label);
  console.log('indexof item in labels', labels.indexOf(menuItems[value]));
  let tabId = labels.indexOf(menuItems[value]);
  if (tabId !== -1) {
    setTabValue(tabId);
  }
  setTabValue(tabId);
  addTab(newValue);
  console.log('newValue add:', newValue);
  console.log('newValue item:', menuItems[newValue]);
  console.log('indexof item in labels', labels.indexOf(menuItems[value]));
};

const addTab = (value) => {
  let id = tabList[tabList.length - 1].id + 1;
  let labels = Object.keys(tabList).map((val) => tabList[val].label);
  console.log('indexof item in labels', labels.indexOf(menuItems[value]));
  let tabId = labels.indexOf(menuItems[value]);

  if (
    labels.indexOf(menuItems[value]) === -1 &&
    labels.length < menuItems.length
  ) {
    setTabList([...tabList, { key: id, id: id, label: menuItems[value] }]);
    console.log('value', value, 'tabList.length', tabList.length);
    setTabValue(tabList.length);
  } else {
    setTabValue(labels.indexOf(menuItems[value]));
  }

  console.log('add tabList length', tabList.length);
```

```
  //let id = tabList[tabList.length - 1].id + 1;
  //setTabList([...tabList, { key: id, id: id }]);
};

const deleteTab = (e) => {
  e.stopPropagation();
  console.log('delete tabList length', tabList.length);
  if (tabList.length === 1) {
    return;
  }
  //let tabId = parseInt(e.target.id);
  let tabId = parseInt(e.currentTarget.id);
  console.log('tabId', tabId);
  let tabIDIndex = 0;

  let tabs = tabList.filter((value, index) => {
    if (value.id === tabId) {
      tabIDIndex = index;
    }
    return value.id !== tabId;
  });

  let currentValue = parseInt(tabValue);
  if (currentValue === tabId) {
    if (tabIDIndex === 0) {
      currentValue = tabList[tabIDIndex + 1].id;
    } else {
      currentValue = tabList[tabIDIndex - 1].id;
    }
  }
  setTabValue(currentValue);
  //setValue(currentValue)
  setTabList(tabs);
};

return (
  <AppBar position="static">
    {console.log('tabValue', tabValue)}
    <Grid container alignItems="center" justify="center">
      <Grid item xl={11} lg={11} md={11} sm={11} xs={11}>
        <Tabs
          value={tabValue}
          onChange={handleTabChange}
          variant="scrollable"
```

```
            scrollButtons="auto"
            style={{ paddingLeft: 176 }}
          >
            {tabList.map((tab) => (
              <Tab
                key={tab.key.toString()}
                value={tab.id}
                label={tab.label}
                icon={tab.id > 0 && <Close id={tab.id} onClick={deleteTab} />}
                className="mytab"
                style={{ color: 'yellow' }}
              />
            ))}
          </Tabs>
        </Grid>
        <Grid item xl={11} lg={11} md={11} sm={11} xs={11}>
          <Tabs
            orientation="vertical"
            variant="scrollable"
            value={value}
            onChange={handleChange}
            aria-label="Vertical tabs example"
          >
            {menuItems.map((tab, i) => (
              <Tab
                key={i}
                value={i}
                label={menuItems[i]}
                className="mytab"
                style={{ color: 'yellow' }}
              />
            ))}
          </Tabs>
        </Grid>
        <Grid item xl={1} lg={1} md={1} sm={1} xs={1}>
          {/*<Button variant="outlined" onClick={addTab}>
            <Add />
            </Button>*/}
        </Grid>
      </Grid>
    </AppBar>
  );
};
```

```
export default DashboardSupply;

_____
import React, { useState } from 'react';
import { AppBar, Tabs, Tab, Grid, Button } from '@material-ui/core';
import Add from '@material-ui/icons/Add';
import Close from '@material-ui/icons/Close';

const menuItems = ['Dashboard', 'Applications', 'Incoming'];

const DashboardSupply = () => {
  const [tabList, setTabList] = useState([
    {
      key: 0,
      id: 0,
      label: 'Dashboard',
    },
  ]);

  const [tabValue, setTabValue] = useState(0);
  const [value, setValue] = React.useState(0);
  const handleTabChange = (event, value) => {
    setTabValue(value);
    setValue(value);
    //handleChange(value);
  };
  const handleChange = (event, newValue) => {
    setValue(newValue);
    addTab(newValue);
    console.log('newValue add:', newValue);
  };
  function a11yProps(index) {
    return {
      id: `vertical-tab-${index}`,
      'aria-controls': `vertical-tabpanel-${index}`,
    };
  }

  const addTab = (value) => {
    let id = tabList[tabList.length - 1].id + 1;
    let labels = Object.keys(tabList).map((val) => tabList[val].label);
    console.log('indexof item in labels', labels.indexOf(menuItems[value]));

    if (
      labels.indexOf(menuItems[value]) === -1 &&
```

```
    labels.length < menuItems.length
  ) {
    setTabList([...tabList, { key: id, id: id, label: menuItems[value] }]);
    setTabValue(value);
  } else {
  }
  //setTabValue(value);
  //setTabValue(labels.indexOf(menuItems[value] + 1));
  menuItems.forEach((item, i) => {
    //console.log('item', item);
    //console.log('foreach indexof', labels.indexOf(item));
    //if (labels.indexOf(item) === -1) {
    //console.log('labels indexof::', labels.indexOf(item));
    //setTabList([...tabList, { key: id, id: id, label: menuItems[value] }]);
    //}
    //if (tabList.indexOf(item) === -1 && tabList.length < menuItems.length)
    //console.log('indexof', tabList.indexOf(item));
    //handleTabChange(i);
  });

  console.log('add tabList length', tabList.length);
  //let id = tabList[tabList.length - 1].id + 1;
  //setTabList([...tabList, { key: id, id: id }]);
};

const deleteTab = (e) => {
  e.stopPropagation();
  console.log('delete tabList length', tabList.length);
  if (tabList.length === 1) {
    return;
  }
  //let tabId = parseInt(e.target.id);
  let tabId = parseInt(e.currentTarget.id);
  console.log('tabId', tabId);
  let tabIDIndex = 0;

  let tabs = tabList.filter((value, index) => {
    if (value.id === tabId) {
      tabIDIndex = index;
    }
    return value.id !== tabId;
  });

  let curValue = parseInt(tabValue);
```

```
  if (curValue === tabId) {
    if (tabIDIndex === 0) {
      curValue = tabList[tabIDIndex + 1].id;
    } else {
      curValue = tabList[tabIDIndex - 1].id;
    }
  }
  setTabValue(curValue);
  //setValue(curValue)
  setTabList(tabs);
};

return (
  <AppBar position="static">
    {console.log(tabValue, 'hohoh')}
    <Grid container alignItems="center" justify="center">
      <Grid item xl={11} lg={11} md={11} sm={11} xs={11}>
        <Tabs
          value={tabValue}
          onChange={handleTabChange}
          variant="scrollable"
          scrollButtons="auto"
          style={{ paddingLeft: 176 }}
        >
          {tabList.map((tab) => (
            <Tab
              key={tab.key.toString()}
              value={tab.id}
              label={tab.label}
              icon={tab.id > 0 && <Close id={tab.id} onClick={deleteTab} />}
              className="mytab"
              style={{ color: 'yellow' }}
            />
          ))}
        </Tabs>
      </Grid>
      <Grid item xl={11} lg={11} md={11} sm={11} xs={11}>
        <Tabs
          orientation="vertical"
          variant="scrollable"
          value={value}
          onChange={handleChange}
          aria-label="Vertical tabs example"
        >
```

```
          {menuItems.map((tab, i) => (
            <Tab
              key={i}
              value={i}
              label={menuItems[i]}
              className="mytab"
              style={{ color: 'yellow' }}
            />
          ))}
        </Tabs>
      </Grid>
      <Grid item xl={1} lg={1} md={1} sm={1} xs={1}>
        {/*<Button variant="outlined" onClick={addTab}>
          <Add />
          </Button>*/}
      </Grid>
    </Grid>
  </AppBar>
 );
};

export default DashboardSupply;
```

```
import React, { useState } from 'react';
import { AppBar, Tabs, Tab, Grid, Button } from '@material-ui/core';
import Add from '@material-ui/icons/Add';
import Close from '@material-ui/icons/Close';

const menuItems = ['Dashboard', 'Applications', 'Incoming'];
/*const menuItems = [
 {
  key: 0,
  id: 0,
  label: 'Dashboard',
 },
 {
  key: 1,
  id: 1,
  label: 'Applications',
 },
 {
  key: 2,
  id: 2,
  label: 'Incoming',
```

```
  },
];*/
const DashboardSupply = () => {
  const [tabList, setTabList] = useState([
    {
      key: 0,
      id: 0,
    },
  ]);

  const [tabValue, setTabValue] = useState(0);
  const [value, setValue] = React.useState(0);
  const handleTabChange = (event, value) => {
    setTabValue(value);
  };
  const handleChange = (event, newValue) => {
    setValue(newValue);
    addTab(newValue);
    console.log('newValue add:', newValue);
  };
  function a11yProps(index) {
    return {
      id: `vertical-tab-${index}`,
      'aria-controls': `vertical-tabpanel-${index}`,
    };
  }

  const addTab = (value) => {
    console.log(' value', value);
    let id = tabList[tabList.length - 1].id + 1;
    tabList.forEach((item) => {
      if (tabList.indexOf(item) !== -1 && tabList.length < menuItems.length)
        setTabList([...tabList, { key: id, id: id }]);
    });
    console.log(tabList.length);
    //let id = tabList[tabList.length - 1].id + 1;
    //setTabList([...tabList, { key: id, id: id }]);
  };

  const deleteTab = (e) => {
    e.stopPropagation();

    if (tabList.length === 1) {
      return;
```

```
  }
  let tabId = parseInt(e.target.id);
  console.log('tabId', tabId);
  let tabIDIndex = 0;

  let tabs = tabList.filter((value, index) => {
    if (value.id === tabId) {
      tabIDIndex = index;
    }
    return value.id !== tabId;
  });

  let curValue = parseInt(tabValue);
  if (curValue === tabId) {
    if (tabIDIndex === 0) {
      curValue = tabList[tabIDIndex + 1].id;
    } else {
      curValue = tabList[tabIDIndex - 1].id;
    }
  }
  setTabValue(curValue);
  //setValue(curValue)
  setTabList(tabs);
};

return (
  <AppBar position="static">
    {console.log(tabValue, 'hohoh')}
    <Grid container alignItems="center" justify="center">
      <Grid item xl={11} lg={11} md={11} sm={11} xs={11}>
        <Tabs
          value={tabValue}
          onChange={handleTabChange}
          variant="scrollable"
          scrollButtons="auto"
        >
          {tabList.map((tab) => (
            <Tab
              key={tab.key.toString()}
              value={tab.id}
              label={menuItems[tab.id]}
              icon={<Close id={tab.id} onClick={deleteTab} />}
              className="mytab"
              style={{ color: 'yellow' }}
```

```
            />
          ))}
        </Tabs>
      </Grid>
      <Grid item xl={11} lg={11} md={11} sm={11} xs={11}>
        <Tabs
          orientation="vertical"
          variant="scrollable"
          value={value}
          onChange={handleChange}
          aria-label="Vertical tabs example"
        >
          {menuItems.map((tab, i) => (
            <Tab
              key={tab}
              value={tab}
              label={menuItems[i]}
              className="mytab"
              style={{ color: 'yellow' }}
            />
          ))}
        </Tabs>
      </Grid>
      <Grid item xl={1} lg={1} md={1} sm={1} xs={1}>
        <Button variant="outlined" onClick={addTab}>
          <Add />
        </Button>
      </Grid>
    </Grid>
  </AppBar>
 );
};

export default DashboardSupply;
```

```
const menuItems = [
 {
  key: 0,
  id: 0,
  label: 'Dashboard',
 },
 {
  key: 1,
  id: 1,
```

```
    label: 'Applications',
  },
  {
    key: 2,
    id: 2,
    label: 'Incoming',
  },
];
```

```
import React, { useState } from 'react';
import { AppBar, Tabs, Tab, Grid, Button } from '@material-ui/core';
import Add from '@material-ui/icons/Add';
import Close from '@material-ui/icons/Close';

const menuItems = ['Dashboard', 'Applications', 'Incoming'];

const DashboardSupply = () => {
  const [tabList, setTabList] = useState([
    {
      key: 0,
      id: 0,
      label: 'Dashboard',
    },
  ]);

  const [tabValue, setTabValue] = useState(0);
  const [value, setValue] = React.useState(0);
  const handleTabChange = (event, value) => {
    setTabValue(value);
    handleChange(value);
  };
  const handleChange = (event, newValue) => {
    setValue(newValue);
    addTab(newValue);
    console.log('newValue add:', newValue);
  };
  function a11yProps(index) {
    return {
      id: `vertical-tab-${index}`,
      'aria-controls': `vertical-tabpanel-${index}`,
    };
  }

  const addTab = (value) => {
```

```
  console.log(' value', value);
  let id = tabList[tabList.length - 1].id + 1;
  tabList.forEach((item, i) => {
    if (tabList.indexOf(item) !== -1 && tabList.length < menuItems.length)
      setTabList([...tabList, { key: id, id: id, label: menuItems[value] }]);
    //handleTabChange(i);
  });
  console.log('add tabList length', tabList.length);
  //let id = tabList[tabList.length - 1].id + 1;
  //setTabList([...tabList, { key: id, id: id }]);
};

const deleteTab = (e) => {
  e.stopPropagation();
  console.log('delete tabList length', tabList.length);
  if (tabList.length === 1) {
    return;
  }
  let tabId = parseInt(e.target.id);
  console.log('tabId', tabId);
  let tabIDIndex = 0;

  let tabs = tabList.filter((value, index) => {
    if (value.id === tabId) {
      tabIDIndex = index;
    }
    return value.id !== tabId;
  });

  let curValue = parseInt(tabValue);
  if (curValue === tabId) {
    if (tabIDIndex === 0) {
      curValue = tabList[tabIDIndex + 1].id;
    } else {
      curValue = tabList[tabIDIndex - 1].id;
    }
  }
  setTabValue(curValue);
  //setValue(curValue)
  setTabList(tabs);
};

return (
  <AppBar position="static">
```

```
{console.log(tabValue, 'hohoh')}
<Grid container alignItems="center" justify="center">
  <Grid item xl={11} lg={11} md={11} sm={11} xs={11}>
    <Tabs
      value={tabValue}
      onChange={handleTabChange}
      variant="scrollable"
      scrollButtons="auto"
    >
      {tabList.map((tab) => (
        <Tab
          key={tab.key.toString()}
          value={tab.id}
          label={tab.label}
          icon={<Close id={tab.id} onClick={deleteTab} />}
          className="mytab"
          style={{ color: 'yellow' }}
        />
      ))}
    </Tabs>
  </Grid>
  <Grid item xl={11} lg={11} md={11} sm={11} xs={11}>
    <Tabs
      orientation="vertical"
      variant="scrollable"
      value={value}
      onChange={handleChange}
      aria-label="Vertical tabs example"
    >
      {menuItems.map((tab, i) => (
        <Tab
          key={i}
          value={i}
          label={menuItems[i]}
          className="mytab"
          style={{ color: 'yellow' }}
        />
      ))}
    </Tabs>
  </Grid>
  <Grid item xl={1} lg={1} md={1} sm={1} xs={1}>
    <Button variant="outlined" onClick={addTab}>
      <Add />
    </Button>
```

```
        </Grid>
      </Grid>
    </AppBar>
  );
};

export default DashboardSupply;
```

```
import React, { useState } from 'react';
import { AppBar, Tabs, Tab, Grid, Button } from '@material-ui/core';
import Add from '@material-ui/icons/Add';
import Close from '@material-ui/icons/Close';

const menuItems = ['Dashboard', 'Applications', 'Incoming'];

const DashboardSupply = () => {
  const [tabList, setTabList] = useState([
    {
      key: 0,
      id: 0,
      label: 'Dashboard',
    },
  ]);

  const [tabValue, setTabValue] = useState(0);
  const [value, setValue] = React.useState(0);
  const handleTabChange = (event, value) => {
    setTabValue(value);
    setValue(value);
    //handleChange(value);
  };
  const handleChange = (event, newValue) => {
    setValue(newValue);
    addTab(newValue);
    console.log('newValue add:', newValue);
  };
  function a11yProps(index) {
    return {
      id: `vertical-tab-${index}`,
      'aria-controls': `vertical-tabpanel-${index}`,
    };
  }

  const addTab = (value) => {
```

```
  let id = tabList[tabList.length - 1].id + 1;
  let labels = Object.keys(tabList).map((val) => tabList[val].label);
  console.log('indexof item in labels', labels.indexOf(menuItems[value]));

  if (
    labels.indexOf(menuItems[value]) === -1 &&
    labels.length < menuItems.length
  ) {
    setTabList([...tabList, { key: id, id: id, label: menuItems[value] }]);
    setTabValue(value);
  }
  setTabValue(value);
  menuItems.forEach((item, i) => {
    //console.log('item', item);
    //console.log('foreach indexof', labels.indexOf(item));
    //if (labels.indexOf(item) === -1) {
    //console.log('labels indexof::', labels.indexOf(item));
    //setTabList([...tabList, { key: id, id: id, label: menuItems[value] }]);
    //}
    //if (tabList.indexOf(item) === -1 && tabList.length < menuItems.length)
    //console.log('indexof', tabList.indexOf(item));
    //handleTabChange(i);
  });

  console.log('add tabList length', tabList.length);
  //let id = tabList[tabList.length - 1].id + 1;
  //setTabList([...tabList, { key: id, id: id }]);
};

const deleteTab = (e) => {
  e.stopPropagation();
  console.log('delete tabList length', tabList.length);
  if (tabList.length === 1) {
    return;
  }
  //let tabId = parseInt(e.target.id);
  let tabId = parseInt(e.currentTarget.id);
  console.log('tabId', tabId);
  let tabIDIndex = 0;

  let tabs = tabList.filter((value, index) => {
    if (value.id === tabId) {
      tabIDIndex = index;
    }
```

```
    return value.id !== tabId;
  });

  let curValue = parseInt(tabValue);
  if (curValue === tabId) {
    if (tabIDIndex === 0) {
      curValue = tabList[tabIDIndex + 1].id;
    } else {
      curValue = tabList[tabIDIndex - 1].id;
    }
  }
  setTabValue(curValue);
  //setValue(curValue)
  setTabList(tabs);
};

return (
  <AppBar position="static">
    {console.log(tabValue, 'hohoh')}
    <Grid container alignItems="center" justify="center">
      <Grid item xl={11} lg={11} md={11} sm={11} xs={11}>
        <Tabs
          value={tabValue}
          onChange={handleTabChange}
          variant="scrollable"
          scrollButtons="auto"
        >
          {tabList.map((tab) => (
            <Tab
              key={tab.key.toString()}
              value={tab.id}
              label={tab.label}
              icon={tab.id > 0 && <Close id={tab.id} onClick={deleteTab} />}
              className="mytab"
              style={{ color: 'yellow' }}
            />
          ))}
        </Tabs>
      </Grid>
      <Grid item xl={11} lg={11} md={11} sm={11} xs={11}>
        <Tabs
          orientation="vertical"
          variant="scrollable"
          value={value}
```

```
            onChange={handleChange}
            aria-label="Vertical tabs example"
          >
            {menuItems.map((tab, i) => (
              <Tab
                key={i}
                value={i}
                label={menuItems[i]}
                className="mytab"
                style={{ color: 'yellow' }}
              />
            ))}
          </Tabs>
        </Grid>
        <Grid item xl={1} lg={1} md={1} sm={1} xs={1}>
          <Button variant="outlined" onClick={addTab}>
            <Add />
          </Button>
        </Grid>
      </Grid>
    </AppBar>
  );
};

export default DashboardSupply;
```
