

Gabriel Ogungbade

Assignment 2

Information Technologies
ITMI23SP

02.02.2025

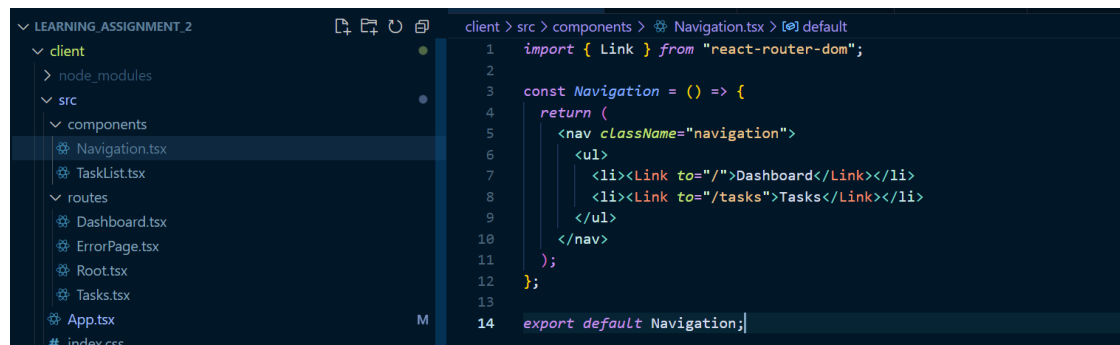
Full stack programming 2025
Miika Reijonen

Assignment report

GitHub: https://github.com/w0daz/learning_assignment_2

1.1. Creating the Navigation component

I created a new components folder in the src directory and added a Navigation.tsx file. I implemented a reusable Navigation component by moving the existing navigation code from the previous implementation into this new component. The component was successfully imported and integrated into the main application structure.



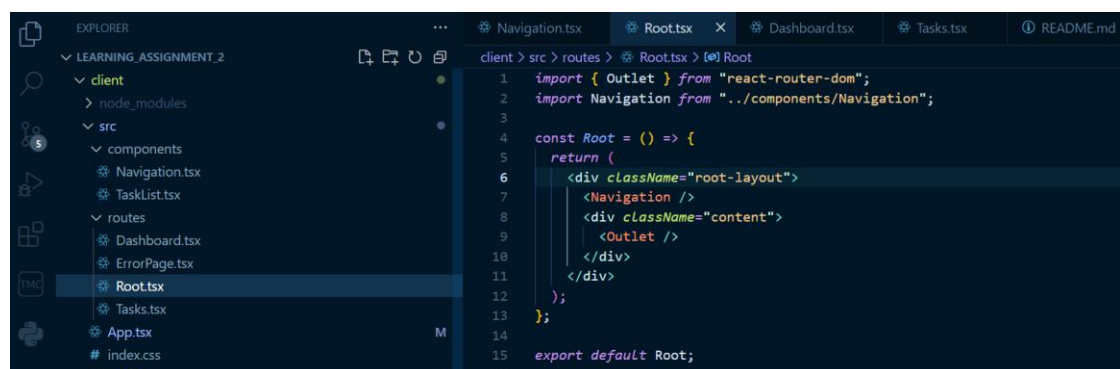
```

client > src > components > Navigation.tsx > [0] default
1  import { Link } from "react-router-dom";
2
3  const Navigation = () => {
4    return (
5      <nav className="navigation">
6        <ul>
7          <li><Link to="/">Dashboard</Link></li>
8          <li><Link to="/tasks">Tasks</Link></li>
9        </ul>
10     </nav>
11   );
12 };
13
14 export default Navigation;

```

1.2. Creating React Router Components

I created a new routes folder in the src directory and implemented two key components: Root.tsx and Dashboard.tsx. The Root component was set up as the base file for the application structure. I successfully moved the existing App component code into the Dashboard component, which now serves as the home view.



```

client > src > routes > Root.tsx > [0] Root
1  import { Outlet } from "react-router-dom";
2  import Navigation from "../components/Navigation";
3
4  const Root = () => {
5    return (
6      <div className="root-layout">
7        <Navigation />
8        <div className="content">
9          <Outlet />
10        </div>
11      </div>
12    );
13 };
14
15 export default Root;

```

1.3. Adding React Router

I installed react-router-dom using npm and verified its addition in package.json. I implemented the Router in the App component with two main routes: one directing to the Root component and another to the Dashboard

component nested within Root. I added the Outlet element to the Root component to handle route rendering.

```

client > src > App.tsx > [default]
1  import { Routes, Route } from 'react-router-dom';
2  import Root from './routes/Root';
3  import Dashboard from './routes/Dashboard';
4  import Tasks from './routes/Tasks';
5  import ErrorPage from './routes/ErrorPage';
6
7  const App = () => {
8    return (
9      <Routes>
10        <Route path="/" element={<Root />} />
11        <Route index element={<Dashboard />} />
12        <Route path="tasks" element={<Tasks />} />
13        <Route path="*" element={<ErrorPage />} />
14      </Routes>
15    );
16  };
17
18
19  export default App;

```

1.4. Adding Route Components

I created two new components: ErrorPage for handling errors and Tasks for managing the task list functionality. I successfully moved the task listing functionality from the Dashboard to the Tasks component and updated the Router configuration to include routes for both new components. The Dashboard was enhanced with a header element.

```

const ErrorPage = () => {
  return (
    <div>
      <h2>404 - Page Not Found</h2>
      <p>Oops! The page you're looking for doesn't exist.</p>
    </div>
  );
};

export default ErrorPage;

```

1.5. Updating Navigation

I relocated the Navigation component to the Root component, positioning it above the Outlet element. I implemented a list of Link elements for navigation between Dashboard and Tasks views. The navigation.tsx code itself was updated to accommodate the change. The layout was updated using Flexbox to create a side menu design, with appropriate styling applied through CSS to achieve the desired visual structure.

```
const Root = () => {  
  return (  
    <div className="root-layout">  
      <Navigation />  
      <div className="content">  
        <Outlet />  
      </div>  
    </div>  
  );  
};
```

```
import { Link } from 'react-router-dom';  
  
const Navigation = () => {  
  return (  
    <nav className="sidebar">  
      <ul className="nav-list">  
        <li>  
          <Link to="/" className="nav-link">Dashboard</Link>  
        </li>  
        <li>  
          <Link to="/tasks" className="nav-link">Tasks</Link>  
        </li>  
      </ul>  
    </nav>  
  );  
};  
  
export default Navigation;
```

1.6. Creating a Task List Component and Using Props

I created a new TaskList.tsx component to handle the task list functionality. I defined a TaskListProps interface to type-check the component's props and implemented the necessary functions for task management. The component

was successfully integrated with the Tasks component, with proper prop passing for the task list and delete functionality.

The implementation of props and component separation improved code reusability and maintenance by clearly defining component responsibilities and data flow.

```
interface TaskListProps {
  taskList: string[];
  onDeleteTask: (task: string) => void;
}

const TaskList = ({ taskList, onDeleteTask }: TaskListProps) => {
  return (
    <ul>
      {taskList.map((task, index) => (
        <li key={index}>
          <span>{task}</span>
          <button onClick={() => onDeleteTask(task)}>Delete</button>
        </li>
      ))}
    </ul>
  );
};

export default TaskList;
```

1.7. Functionality Test & Overview

