Державний вищий навчальний заклад

Ужгородський національний університет

Факультет інформаційних технологій.

ЛАБОРАТОРНА РОБОТА №5

Тема: “**Реалізація фронтед частини. Ч2 - маршрутизація**
”

Виконав студент II курсу

спеціальності “Інженерія

програмного
забезпечення.”

Попович Ернест Олегович

Ужгород 2025

Мета: Навчитися використовувати react-router у проєкті

Завдання:

1. Підключити react-router або @tanstack/react-router

2. Реалізувати маршрутизацію у прєкті (користувач повинен мати можливість навігації між сторінками)

3. Створити пул-реквест

4. Оформити відповідного взірця звіт

Хід виконання лаболаторної роботи

Для того, щоб почати використовувати React-router потрібно завантажити відпровідну бібліотеку командою:

```
H:~/Documents/pract/React/react--projects/soundcloud-project$ npm install react-router
```

Далі потрібно огорнути весь проект в <BrowseRouter/> тег, на рівні App.tsx або main.tsx. Одночасно і в App і в main обгортати в цей тег неможна:

Рис.1 main.tsx:

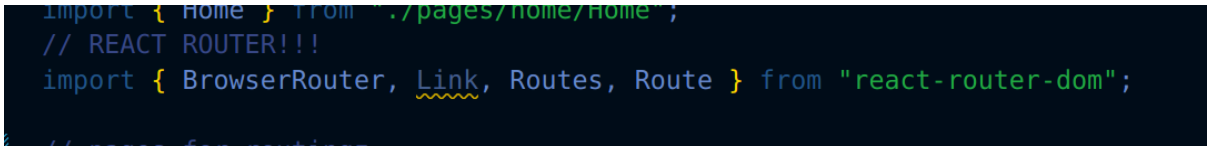
```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import { Provider } from 'react-redux'
import SoundCloud from '../compontents/SoundCloud.tsx'

import { BrowserRouter } from 'react-router-dom'
//
import {store} from "../redux/storages/store.ts"
// import { createBrowserRouter } from 'react-router-dom'

// const router = createBrowserRouter([]);
createRoot(document.getElementById('root')!).render(
  <StrictMode>
    <BrowserRouter>

    <Provider store = {store}>

    <SoundCloud />

    </Provider>

    </BrowserRouter>
  </StrictMode>,
)
```

Або:

Рис. 2 SoundCloud.tsx (App.tsx):

```
function SoundCloud() {
  return (
    <>
      <BrowserRouter>
        {/* scroll to top after virtual reloading the pages */}
        <ScrollToTop></ScrollToTop>
        <TitleChange></TitleChange>
        <main
          className={` sm:w-[40rem] md:w-[55rem] lg:w-[70rem]  place-self-cen
        >
          <section
            className={`w-[100%] flex align-center place-self-center relative
          >
            <Navbar></Navbar>
          </section>
          <Routes>
            <Route path="/" element={<Home />} /> {/* стартова сторінка */}
            {buttsNavbar.map((item, index) => (
              <Route
                key={index}
                path={item.path}
                element={<item.component />}
              />
            ))}
            <Route path="/copyright" element={<Copyright />}></Route>
          </Routes>
          <Player></Player>
        </main>
      </BrowserRouter>
    </>
  );
}
```

Далі, для роботи з Router, потрібно імпортувати такі елементи, як: Router, Route, Link (не завжди):



```
import { Home } from "./pages/home/Home";
// REACT ROUTER!!!
import { BrowserRouter, Link, Routes, Route } from "react-router-dom";
```

В самій jsx розмітці потрібно огорнути в <Router/> теги <Route/> які якраз відповідають за маршрутизацію до інших компонентів:

```
<Routes>
  <Route path="/" element={<Home />} /> {/* home start page */}
  {buttsNavbar.map((item, index) => (
    <Route
      key={index}
      path={item.path}
      element={<item.component />}
    />
  ))}
  <Route path="/copyright" element={<Copyright />}></Route>
</Routes>
```

Тут кожен маршрут (Route) має відповідні атрибути, які необхідні для перенаправлення на відповідін сторінки:

Path - атрибут, який містить в собі адрес до сторінки (може бути ‘/’ як стартовий адрес, може бути `/home,  /about` і так далі).

Element - це компонент, який буде відображатися відповідно до адресу який зображений в полі протоколу https. Тобто стартова сторінка (це пустий хост , ‘/’) буде відображати <Home/> page:

```
<Route path="/" element={<Home />} /> {/* home start page  */}
```

Для оптимізації я обрав стратегію DRY-принципу, тому у всьому проекту у мене мінімальне дублювання коду. Ось прямо в <Routes>:

```
{buttsNavbar.map((item, index) => (
  <Route
    key={index}
    path={item.path}
    element={<item.component />}
  />
))}
```

Для оптимізації я вирішив винести в окремий файл об’єкт, який містить всі компоненти та шляхи (paths) для маршрутизації:

```
export const buttsNavbar = [
  { content: "Home", component: Home, path: "/discover" },
  { content: "Feed", component: Feed, path: "/feed" },
  { content: "Library", component: Library, path: "/you/*" },
  { content: "Try Artist Pro", component: TryArtist,path: "/try" },
  { content: "For Artists", component:ForArtists, path: "/artist" },
  { content: "Upload", component: Upload, path: "/upload" },
];
```

В проекті не один Router, їх два (це Library сторінка, як в SoundCloud):

```
<AnimatePresence>
<motion.main initial = {{opacity: 0, y: 0}} animate = {{opacity:1, y: 10}} className="flex

    <ButtonsTitle></ButtonsTitle>
     <Routes>
    <Route index  element = {<OverviewPage/>}></Route>

    <Route index  element = {<Navigate to="library"/>}></Route>
        {libraryPages.map((item,index) => (
            <Route key={index} path ={item.path} element = {<item.component/>}></Route>
        ))}
     </Routes>

    <UndergroundResources></UndergroundResources>
</motion.main>
    </AnimatePresence>
</>);
};
```

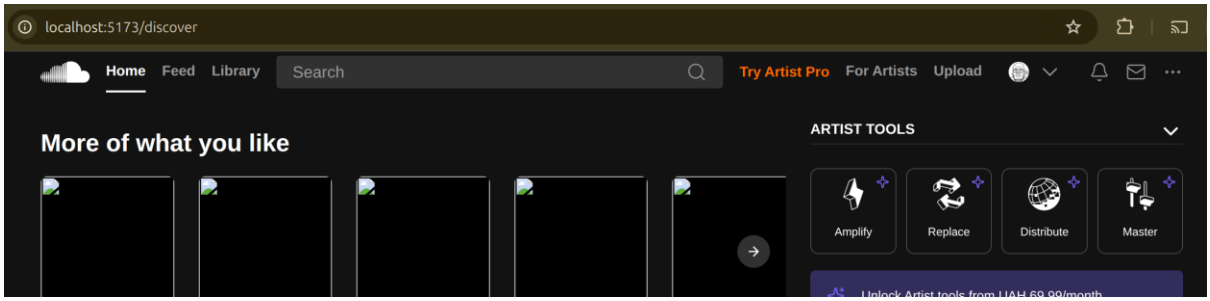
LibraryPages:

```
export const libraryPages = [
    {content: "Overwiew",isSelected: false, component: OverviewPage, path:"library"},
    {content: "Likes", isSelected: false, component:LikesPage, path: "likes"},
    {content: "Playlists", isSelected: false, component: PlaylistsPage, path: "sets"},
    {content: "Albums", isSelected: false, component: AlbumsPage, path: "albums"},
    {content: "Stations", isSelected:false, component: StationsPage, path:"stations"},
    {content: "Following", isSelected:false, component: FollowingPage, path: "following"},
    {content: "History", isSelected: false, component: HistoryPage, path: "history"},
]
```

Якщо для маршрутизації не використовується елемент <Link/>, то з Router можна імпортувати хук useNavigate, який дозволить безперешкодно перенаправляти сторінку по path (Реаліловано через Map):

```
const dispatch = useNavbarAppDispatch();
const navigate = useNavigate();

const [isInputOnFocus, setIsInputOnFocus] = useState<boolean>(false);
const [isButtonActive, setIsButtonActive] = useState(Array.from({length: 6},
const routeNavbarMap = new Map();
for (let i = 0; i < buttsNavbar.length; i++) {
  routeNavbarMap.set(buttsNavbar[i].component, buttsNavbar[i].path);
}
const handleNavbarRouting = (content: React.ComponentType, id: number) => {
  setIsButtonActive((prev) => prev.map((_,index) => ({isActive:index == id}))
  const path = routeNavbarMap.get(content);
  if (path) {
    navigate(path);
  }
    // setTimeout(() => window.location.reload(), 0);
  dispatch(setAllPopupWindowClose());
};
```

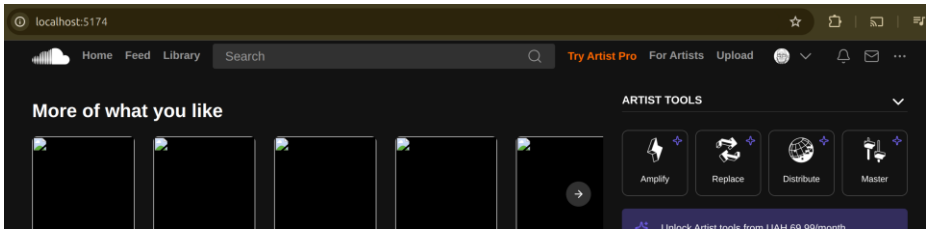
Ось головна сторінка:



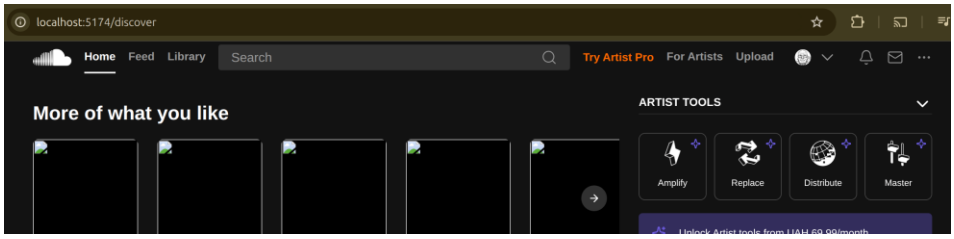
В полі адресу вказано /discover, а це <Home/> сторінка, але при першому запуску Router не зможе відобразити цей /discover, тому Home треба відобразити, коли ‘/’.

Коли натискається кнопка на Home, то тоді до URL додається discovery:

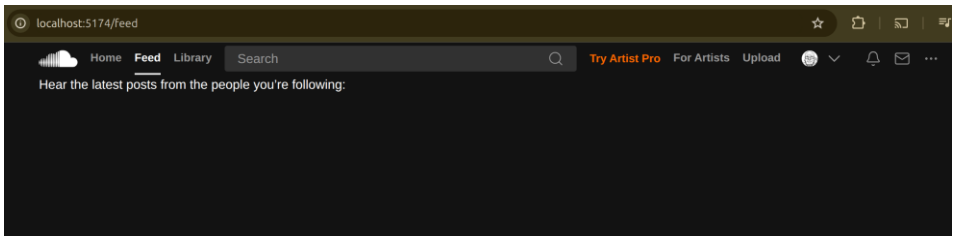
На початку:



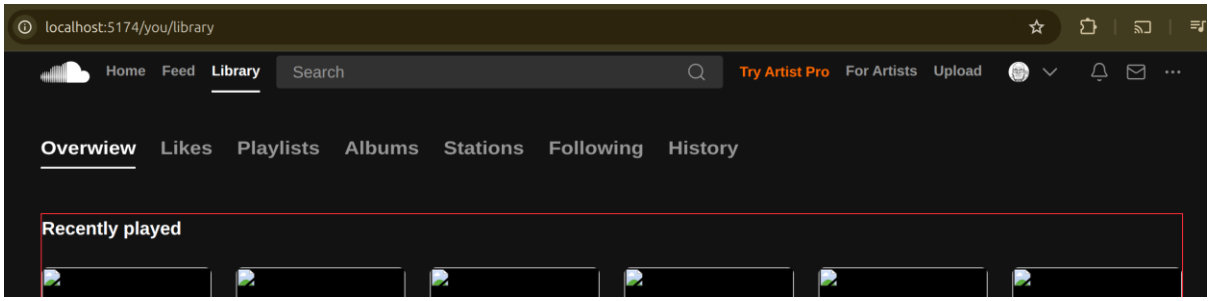
Після натиснення на Home кнопку в navbar (з’явився discovery):



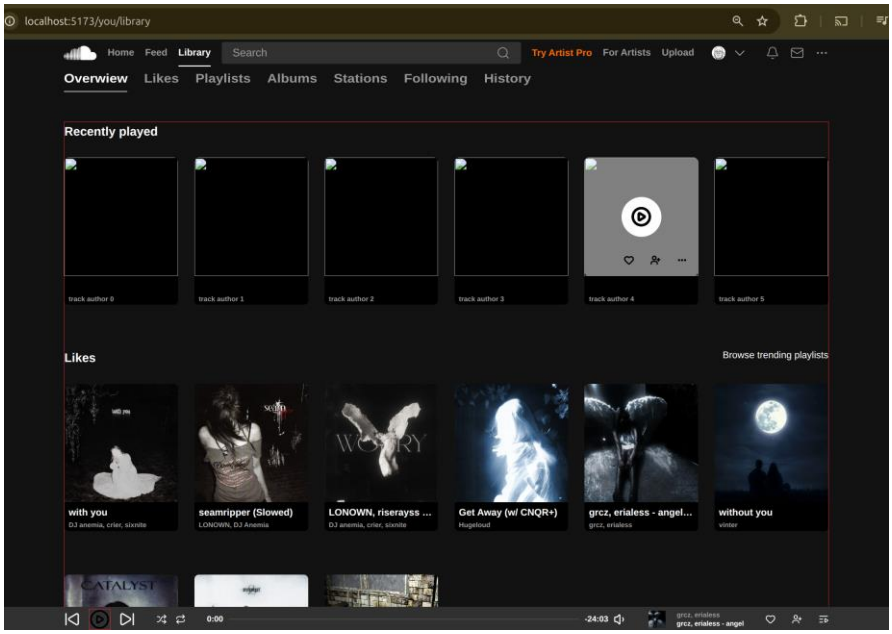
Сторінка Feed:



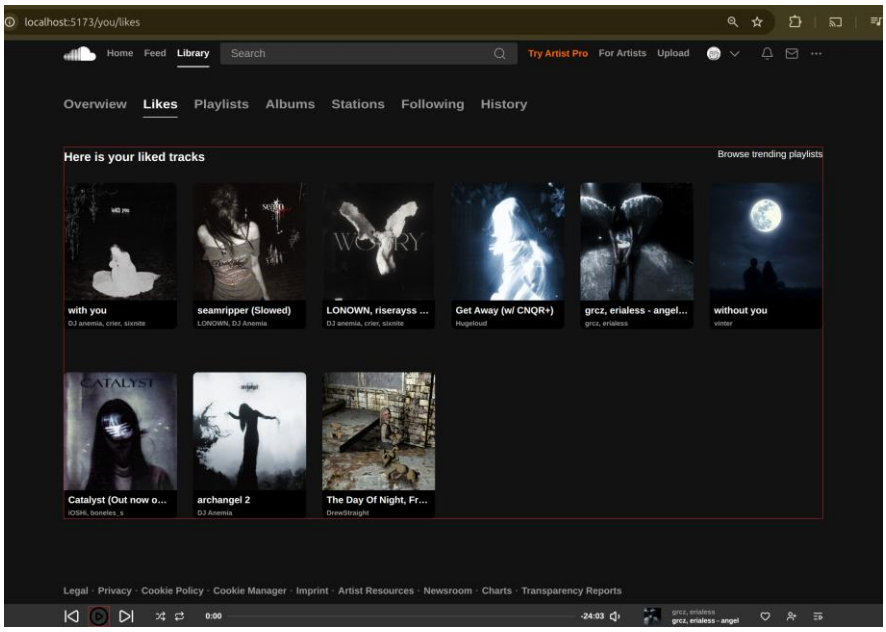
Сторінка Library:



А в library є теж Router по library сторінкам. Overwiew:



Likes сторінка:



Створив pull Request:

Висновок:

В даній лаболаторній роботі було інтегровано в проект бібліотеку React Router, яка дає можливість динамічно переміщатися між сторінками. Було завантажено відповідні бібліотеки: react-router і react-router-dom.

Для того, щоб він працював, потрібно обгорнути проект в <BrowserRouter/> тег або на рівні компонентів або на рівні комонента-рендера (main.tsx). Два таких тега будь де в проекті викличе помилку (you cant use <Router> in <Router>).

Для маршрутизації, потрібно іипортувати такі елементи, як Routes, Route і Link або useNavigate. В моєму випадку я використав useNavigate, так як він більш гнучкий.

В проекті не одна реалізація Router, їх дві: для глобальних сторінок і сторінок бібліотеки.