Babyboo - Kode index

[**1. Forsider**](#_grna4vypskj9) **1**

[**2. Backend**](#_sxi3128i22o5) **12**

[**3. Database**](#_dyyyj23exvay) **129**

[**4. Api**](#_38t3n0dfgrys) **177**

# **Forsider**

---

import Layout from '../layouts/Layout.astro';

import '../../assets/scss/astro-ecommerce.scss';

import ComplexNavbar from '../components/complexNavbar';

import Sidebar from '../components/sidebar';

const links = [

{

text: 'Profil',

component: 'EditUser',

icon: 'mdi mdi-contacts menu-icon'

},

{

text: 'Produkter',

icon: 'mdi mdi-archive-arrow-up',

submenu: [

{

text: 'Produkter',

component: 'ProductPanel',

icon: 'mdi mdi-archive-arrow-up',

},

{

text: 'Tilføj Produkt',

component: 'Addproduct',

icon: 'mdi mdi-plus-circle'

},

]

},

{

text: 'Kategorier',

icon: 'mdi mdi-list-box-outline',

submenu: [

{

text: 'Kategorier',

component: 'CategoryTable',

icon: 'mdi mdi-list-box-outline',

},

{

text: 'Tilføj Kategorier',

component: 'AddCategory',

icon: 'mdi mdi-plus-circle'

}

]

},

];

---

<Layout title="Adminstratorpanel">

<main>

<div id="navigation" style="margin-left: 250px;">

<ComplexNavbar client:only />

</div>

<div id="sidebar">

<Sidebar links={links} client:only />

</div>

</main>

</Layout>

---

import Layout from '../layouts/Layout.astro';

import '../../assets/scss/astro-ecommerce.scss';

import ComplexNavbar from '../components/complexNavbar';

import ComplexFooter from '../components/complexFooter';

import Sidebar from '../components/sidebar';

const links = [

{

text: 'Profil',

component: 'EditUser',

icon: 'mdi mdi-contacts menu-icon'

},

];

---

<Layout title="Brugerpanel">

<main>

<div id="navigation" style="margin-left: 250px;">

<ComplexNavbar client:only />

</div>

<div id="sidebar">

<Sidebar links={links} client:only />

</div>

<div id="footer" style="margin-left: 250px;">

<ComplexFooter client:only />

</div>

</main>

</Layout>

---

import Layout from '../layouts/Layout.astro';

import '../../assets/scss/astro-ecommerce.scss';

import ComplexNavbar from '../components/complexNavbar';

import CheckoutOrderSummary from '../components/checkout/checkoutOrderSummary';

import ComplexFooter from '../components/complexFooter';

---

<Layout title="Kurv">

<main>

<ComplexNavbar client:only />

<h3 style="text-align: center; margin-top: 10px; font-weight: bolder;">Betalings form</h3>

<CheckoutOrderSummary client:only/>

<div style="margin-top: 40px;">

<ComplexFooter client:only />

</div>

</main>

</Layout>

---

import Layout from '../layouts/Layout.astro';

import '../../assets/scss/astro-ecommerce.scss';

import ComplexNavbar from '../components/complexNavbar';

import ComplexFooter from '../components/complexFooter';

import PromoSectionLarge from '../components/promo/promoSectionLarge';

import CardCategory from '../components/products/cardCategory';

import IncentiveCols from '../components/incentives/incentiveCols';

import ProductDisplay from '../components/products/categoryFilters';

import PageVisitTracker from "../components/NotifcationSystem/PageVisitTracker"

import CookieBannerModal from "../components/thirdparty/CookieBanner"

---

<Layout title="Forside">

<main>

<ComplexNavbar client:only />

<PageVisitTracker client:only/>

<CookieBannerModal client:only/>

<div class="my-3 d-flex justify-content-between">

<div class="flex-grow-1 me-3">

<PromoSectionLarge bannernumber={1} />

</div>

<div class="flex-grow-1">

<PromoSectionLarge bannernumber={2} />

</div>

</div>

<div>

<CardCategory client:only

/>

</div>

<div class="mb-5">

<IncentiveCols />

</div>

<div id="productsection">

<ProductDisplay client:only/>

</div>

<ComplexFooter client:only />

</main>

</Layout>

---

import Layout from '../layouts/Layout.astro';

import '../../assets/scss/astro-ecommerce.scss';

import ComplexNavbar from '../components/complexNavbar';

import ComplexFooter from '../components/complexFooter';

import Categories from '../components/products/Categories';

import {UserRoles} from '../../FrontendRequests/Enums/Usertypes.js'

---

<Layout title="Kategorier">

<main>

<ComplexNavbar client:only />

<h2 style="text-align: center;">Kategorier</h2>

<Categories client:only/>

<ComplexFooter client:only />

</main>

</Layout>

---

import Layout from '../layouts/Layout.astro';

import '../../assets/scss/astro-ecommerce.scss';

import ComplexNavbar from '../components/complexNavbar';

import ComplexFooter from '../components/complexFooter';

import KontaktPlade from '../components/contacts/kontaktPlade';

import KontaktFormular from '../components/contacts/kontaktformular';

import {UserRoles} from '../../FrontendRequests/Enums/Usertypes.js'

---

<Layout title="Kontakt">

<main>

<ComplexNavbar client:only />

<KontaktPlade />

<div class="mx-auto text-center w-md-60 mb-5">

<h3 style="padding-top: 39px;">Skriv til os</h3>

</div>

<KontaktFormular client:only/>

<ComplexFooter client:only />

</main>

</Layout>

---

import Layout from '../layouts/Layout.astro';

import '../../assets/scss/astro-ecommerce.scss';

import ComplexNavbar from '../components/complexNavbar';

import ShoppingCart from '../components/cart/shoppingCart';

import {UserRoles} from '../../FrontendRequests/Enums/Usertypes.js'

import ComplexFooter from '../components/complexFooter';

---

<Layout title="kurv">

<main>

<ComplexNavbar client:only />

<ShoppingCart client:only/>

<div style="margin-top: 40px;">

<ComplexFooter client:only />

</div>

</main>

</Layout>

---

import Layout from '../layouts/Layout.astro';

import '../../assets/scss/astro-ecommerce.scss';

import ComplexNavbar from '../components/complexNavbar';

import ComplexFooter from '../components/complexFooter';

import Loginformular from '../components/profile/login';

import {UserRoles} from '../../FrontendRequests/Enums/Usertypes.js'

---

<Layout title="Login">

<main>

<ComplexNavbar client:only />

<div class="mx-auto text-center w-md-60 mb-5">

<h3 style="padding-top: 39px;">Login</h3>

</div>

<Loginformular client:only/>

<ComplexFooter client:only />

</main>

</Layout>

---

import Layout from '../layouts/Layout.astro';

import '../../assets/scss/astro-ecommerce.scss';

import ComplexNavbar from '../components/complexNavbar';

import ComplexFooter from '../components/complexFooter';

import Logudformular from '../components/profile/logud';

import {UserRoles} from '../../FrontendRequests/Enums/Usertypes.js'

---

<Layout title="Logud">

<main>

<ComplexNavbar client:only />

<div class="mx-auto text-center w-md-60 mb-5">

<h3 style="padding-top: 39px;">Login</h3>

</div>

<Logudformular client:only/>

<ComplexFooter client:only />

</main>

</Layout>

---

import Layout from '../layouts/Layout.astro';

import '../../assets/scss/astro-ecommerce.scss';

import ComplexNavbar from '../components/complexNavbar';

import ComplexFooter from '../components/complexFooter';

import {UserRoles} from '../../FrontendRequests/Enums/Usertypes.js'

import Sidebar from '../components/sidebar';

const links = [

{

text: 'Profil',

component: 'EditUser',

icon: 'mdi mdi-contacts menu-icon'

},

{

text: 'Produkter',

icon: 'mdi mdi-archive-arrow-up',

submenu: [

{

text: 'Produkter',

component: 'ProductPanel',

icon: 'mdi mdi-archive-arrow-up',

},

{

text: 'Tilføj Produkt',

component: 'Addproduct',

icon: 'mdi mdi-plus-circle'

},

]

},

{

text: 'Kategorier',

icon: 'mdi mdi-list-box-outline',

submenu: [

{

text: 'Kategorier',

component: 'CategoryTable',

icon: 'mdi mdi-list-box-outline',

},

{

text: 'Tilføj Kategorier',

component: 'AddCategory',

icon: 'mdi mdi-plus-circle'

}

]

},

];

---

<Layout title="Medarbejderpanel">

<main>

<div id="navigation" style="margin-left: 250px;">

<ComplexNavbar client:only />

</div>

<div id="sidebar">

<Sidebar links={links} client:only />

</div>

</main>

</Layout>

---

import Layout from '../layouts/Layout.astro';

import '../../assets/scss/astro-ecommerce.scss';

import ComplexNavbar from '../components/complexNavbar';

import ComplexFooter from '../components/complexFooter';

import OrderSummaries from '../components/order/orderSummaries';

---

<Layout title="Ordre oversigt">

<main>

<ComplexNavbar client:only />

<h3 style="text-align: center; margin-top: 10px; font-weight: bolder;">Checkout Forms</h3>

<OrderSummaries order={null} isadmin={false} client:only/>

<div style="margin-top: 40px;">

<ComplexFooter client:only />

</div>

</main>

</Layout>

---

import Layout from '../layouts/Layout.astro';

import '../../assets/scss/astro-ecommerce.scss';

import ComplexNavbar from '../components/complexNavbar';

import ComplexFooter from '../components/complexFooter';

import KontaktPlade from '../components/contacts/kontaktPlade';

import Privatpolitiks from '../components/store/privatpolitik';

---

<Layout title="Privatpolitik">

<main>

<ComplexNavbar client:only />

<KontaktPlade />

<Privatpolitiks client:only/>

<ComplexFooter client:only />

</main>

</Layout>

---

import Layout from '../layouts/Layout.astro';

import '../../assets/scss/astro-ecommerce.scss';

import ComplexNavbar from '../components/complexNavbar';

import ComplexFooter from '../components/complexFooter';

import ProductOverviewGallery from '../components/products/productOverviewGallery';

---

<Layout title="Produkt detaljier">

<main>

<ComplexNavbar client:only />

<div class="my-5 product-overview-container">

<ProductOverviewGallery client:only/>

</div>

<ComplexFooter client:only />

</main>

</Layout>

<style>

.product-overview-container {

max-width: 80%; /\* Adjust the width to control how small the section is \*/

margin: 0 auto; /\* Center the container on the page \*/

padding: 20px; /\* Add some padding inside the container \*/

border: 1px solid #ddd; /\* Add a border to create the 'edges' \*/

border-radius: 10px; /\* Optional: Add rounded corners \*/

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); /\* Optional: Add shadow for a card-like effect \*/

background-color: #fff; /\* Ensure a light background to contrast with the border \*/

}

</style>

---

import Layout from '../layouts/Layout.astro';

import '../../assets/scss/astro-ecommerce.scss';

import ComplexNavbar from '../components/complexNavbar';

import ComplexFooter from '../components/complexFooter';

import Registerformular from '../components/profile/register';

---

<Layout title="Register">

<main>

<ComplexNavbar client:only />

<div class="mx-auto text-center w-md-60 mb-5">

<h3 style="padding-top: 39px;">Register</h3>

</div>

<Registerformular client:only/>

<ComplexFooter client:only />

</main>

</Layout>

# **Backend**

---

import { Folders, Identifier } from '../../FrontendRequests/Enums/Images';

export interface Props {

title: string;

}

const { title } = Astro.props;

---

<!DOCTYPE html>

<html lang="en">

<head>

<link href="https://fonts.googleapis.com/css2?family=Inter:wght@400;500;600&display=swap" rel="stylesheet">

<link rel="stylesheet" href="/assets/scss/globals.css" />

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width" />

<link rel="icon" type="image/svg+xml" href={`https://res.cloudinary.com/${import.meta.env.CLOUDINARY\_CLOUD\_NAME}/image/upload/v${new Date().getTime()}/${Folders.MEDIA}/${Identifier.FAVICON}.svg`} />

<link

href="https://fonts.googleapis.com/css?family=Open+Sans:300,400,600,700|Noto+Sans:300,400,500,600,700,800|PT+Mono:300,400,500,600,700"

rel="stylesheet"

/>

<meta name="generator" content={Astro.generator} />

<title>{title}</title>

<link rel="canonical" href="https://www.creative-tim.com/astro/" />

</head>

<body style={{ backgroundColor: '#f0f0f0' }}>

<!-- Google Tag Manager (noscript) -->

<noscript><iframe src="https://www.googletagmanager.com/ns.html?id=GTM-NKDMSK6"

height="0" width="0" style="display:none;visibility:hidden"></iframe></noscript>

<!-- End Google Tag Manager (noscript) -->

<slot />

</body>

<!-- Bootstrap JavaScript Bundle with Popper -->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdiJJigb/j/68SIy3Te4Bkz" crossorigin="anonymous"></script>

<!-- apexcharts bundle -->

<script src="https://cdn.jsdelivr.net/npm/apexcharts" crossorigin="anonymous"></script>

<!-- Product JavaScript -->

<script>

import "/assets/js/astro-ecommerce.js"

</script>

</html>

<style is:global>

code {

font-family: Menlo, Monaco, Lucida Console, Liberation Mono, DejaVu Sans Mono,

Bitstream Vera Sans Mono, Courier New, monospace;

}

</style>

'use client';

import React from 'react';

import { Navbar, Container, Nav } from 'react-bootstrap';

import { NotificationBasket } from './NotifcationSystem/NotificationBasket';

import { NavbarLogin } from './profile/navbarlogin';

import { Folders, Identifier } from '../../FrontendRequests/Enums/Images';

const ComplexNavbar = () => {

// Define styles with the correct type

const navButtonStyle: { base: React.CSSProperties; hover: React.CSSProperties } = {

base: {

backgroundSize: '200% 200%',

color: '#333',

fontSize: '1rem',

padding: '0.5rem 0.8rem',

textDecoration: 'none',

whiteSpace: 'nowrap',

position: 'relative',

transition: 'background-position 0.4s ease, color 0.4s ease',

},

hover: {

backgroundPosition: 'right center',

color: `${import.meta.env.LIGHT\_PURPLE}`,

},

};

return (

<Navbar

bg="white"

expand="lg"

className="border-bottom"

style={{

padding: '0.5rem 1rem',

boxShadow: '0 2px 4px rgba(0,0,0,0.1)',

}}

>

<Container fluid>

{/\* Logo \*/}

<Navbar.Brand href="/" style={{ textDecoration: 'none' }}>

<img

src={`https://res.cloudinary.com/${import.meta.env.CLOUDINARY\_CLOUD\_NAME}/image/upload/v${new Date().getTime()}/${Folders.MEDIA}/${Identifier.NAVBAR}.jpg`}

alt="Navbar Logo"

width={120}

height={70}

style={{ objectFit: 'contain' }}

/>

</Navbar.Brand>

{/\* Center Navigation \*/}

<Nav className="w-100 justify-content-center text-center">

<div className="d-flex flex-wrap justify-content-center align-items-center">

{[

{ label: 'Butik', link: '/Butik' },

{ label: 'Kategorier', link: '/Kategorier' },

{ label: 'Privatpolitik', link: '/Privatpolitik' },

].map((item, index) => (

<React.Fragment key={index}>

<a

href={item.link}

style={navButtonStyle.base}

onMouseEnter={(e) =>

Object.assign(e.currentTarget.style, navButtonStyle.hover)

}

onMouseLeave={(e) =>

Object.assign(e.currentTarget.style, navButtonStyle.base)

}

>

{item.label}

</a>

{index < 4 && (

<span className="mx-1 text-dark d-none d-md-inline">|</span>

)}

</React.Fragment>

))}

</div>

</Nav>

{/\* Right Icons \*/}

<Nav className="ms-auto">

<NotificationBasket />

<NavbarLogin />

</Nav>

</Container>

</Navbar>

);

};

export default ComplexNavbar;

import React, { useState, useEffect, type JSX } from 'react';

import ProductPanel from './products/productspanel';

import Addproduct from '../components/products/addproduct';

import EditProduct from '../components/products/editProduct';

import CategoryTable from '../components/category/categoryTable';

import AddCategory from '../components/category/addCategory';

import EditUser from './profile/profile';

import {type User, IsLoggedIn, fetchUserById} from "../../FrontendRequests/Requests-Api/User"

interface Link {

text: string;

icon: string;

component?: string;

badge?: string;

submenu?: Link[];

}

interface Props {

links: Link[];

}

type ComponentMap = {

[key: string]: (props?: any) => JSX.Element;

};

const components: ComponentMap = {

EditUser,

ProductPanel,

Addproduct,

EditProduct,

CategoryTable,

AddCategory,

};

const Sidebar = ({ links }: Props) => {

const [activeIndex, setActiveIndex] = useState<{ mainIndex: number | null; subIndex: number | null }>({ mainIndex: null, subIndex: null });

const [activeComponent, setActiveComponent] = useState<JSX.Element | null>(null);

const [openDropdownIndex, setOpenDropdownIndex] = useState<number | null>(null);

const [LoggedIn, setIsLoggedIn] = useState<boolean | null>(null);

const [User, SetUser] = useState<User | null>(null);

const [loginChecked, setLoginChecked] = useState(false);

const checkLoginStatus = async () => {

try {

const result = await IsLoggedIn();

setIsLoggedIn(result.result);

if (result.result) {

const userresult = await fetchUserById(result.data.user.userid)

SetUser(userresult.data)

}

} catch (error) {

console.error('Error checking login status:', error);

setIsLoggedIn(false);

}

};

useEffect(() => {

const initialize = async () => {

await checkLoginStatus();

setLoginChecked(true);

};

initialize();

}, []); // Empty dependency array to ensure this only runs once on mount

useEffect(() => {

if (loginChecked && User !== null) {

setActiveComponent(<EditUser Userprofile={User} isadmin={false} />)

setActiveIndex({ mainIndex: 0, subIndex: null }); // Set active index to the first link

}

}, [loginChecked, User]); // This only triggers once both conditions are met

const handleMainClick = (component: string | undefined, index: number) => {

if (component) {

const ComponentElement = components[component];

if (ComponentElement === EditUser) {

if (User !== null) {

setActiveComponent(<EditUser Userprofile={User} isadmin={false} />)

}

}

}

setActiveIndex({ mainIndex: index, subIndex: null });

};

const handleSubClick = (component: string | undefined, mainIndex: number, subIndex: number) => {

if (component) {

const ComponentElement = components[component];

setActiveComponent(<ComponentElement />);

}

setActiveIndex({ mainIndex, subIndex });

};

const toggleDropdown = (index: number) => {

setOpenDropdownIndex(openDropdownIndex === index ? null : index);

};

return (

<div>

<style>{`

.sidebar {

background-color: #343a40;

color: #ffffff;

width: 250px;

padding-top: 1rem;

top: 0;

position: fixed; /\* Make the sidebar fixed \*/

height: 100vh; /\* Full height \*/

}

.sidebar .nav-link {

padding: 10px;

font-size: 14px;

border-radius: 10px;

}

.sidebar .nav-link:not(.active) {

color: #ffffff;

}

.sidebar .nav-link.active {

background-color: ${import.meta.env.LIGHT\_BLUE};

color: #ffffff;

}

.sidebar .nav-link i {

font-size: 18px;

color: #ffffff;

}

.submenu {

padding-left: 20px;

background-color: #2e3b4e;

}

.submenu .nav-link.active {

background-color: ${import.meta.env.LIGHT\_RED};

color: #ffffff;

}

`}</style>

{LoggedIn ? (

<>

<nav className="sidebar">

<div className="d-flex align-items-center p-3">

<img src="images/avatar.jpg" alt="Profile" className="rounded-circle me-2" style={{ width: '40px', height: '40px' }} />

<div>

<div className="text-white fw-bold">{User?.username}</div>

<div className="text-muted">@{User?.member\_type}</div>

</div>

</div>

<ul className="nav flex-column">

{links.map((link, index) => (

<li className="nav-item" key={index}>

{link.submenu ? (

<>

<a

href="#"

className={`nav-link d-flex justify-content-between align-items-center px-3 ${activeIndex.mainIndex === index ? 'active' : ''}`}

onClick={() => {

toggleDropdown(index);

handleMainClick(link.component, index);

}}

>

<span>

<i className={`mdi ${link.icon} me-2`} />

{link.text}

</span>

<i className={`mdi mdi-chevron-${openDropdownIndex === index ? 'up' : 'down'}`} />

</a>

{openDropdownIndex === index && (

<ul className="nav flex-column submenu">

{link.submenu.map((sublink, subindex) => (

<li className="nav-item" key={subindex}>

<a

href="#"

className={`nav-link ${activeIndex.mainIndex === index && activeIndex.subIndex === subindex ? 'active' : ''} text-white`}

onClick={() => handleSubClick(sublink.component, index, subindex)}

>

<i className={`mdi ${sublink.icon} me-2`} />

{sublink.text}

</a>

</li>

))}

</ul>

)}

</>

) : (

<a

href="#"

className={`nav-link d-flex justify-content-between align-items-center px-3 ${activeIndex.mainIndex === index ? 'active' : ''}`}

onClick={() => handleMainClick(link.component, index)}

>

<span>

<i className={`mdi ${link.icon} me-2`} />

{link.text}

</span>

{link.badge && <span className="badge">{link.badge}</span>}

</a>

)}

</li>

))}

</ul>

</nav>

<div style={{ marginLeft: "250px", padding: "20px" }}>

{activeComponent}

</div>

</>

) : (

<div>

<nav className="sidebar">

<div className="d-flex align-items-center p-3">

<img src="images/avatar.jpg" alt="Profile" className="rounded-circle me-2" style={{ width: '40px', height: '40px' }} />

</div>

</nav>

<div className='text-center' style={{ marginLeft: "250px", padding: "20px" }}>

Unathorized

</div>

</div>

)}

</div>

);

}

export default Sidebar;

import { Container, Row, Col, Form, InputGroup, Button } from "react-bootstrap";

import type { Settings } from './thirdparty/CookieBanner';

import { useEffect, useState, type SetStateAction } from "react";

export default function ComplexFooter() {

const [email, setEmail] = useState("");

const handleEmailChange = (e: { target: { value: SetStateAction<string>; }; }) => {

setEmail(e.target.value);

};

useEffect(() => {

// Dynamically load the Trustpilot script

const script = document.createElement('script');

script.type = 'text/javascript';

script.src = '//widget.trustpilot.com/bootstrap/v5/tp.widget.bootstrap.min.js';

script.async = true;

// Append the script to the document

document.body.appendChild(script);

// Cleanup script when the component unmounts

return () => {

document.body.removeChild(script);

};

}, []);

return (

<footer className="w-full py-5 bg-white">

<Container>

{/\* Footer Columns \*/}

<Row className="mb-4">

{/\* Gallery Column \*/}

<Col md={6} lg={3} className="mb-4">

<h3 className="text-lg font-semibold text-gray-900">Kontakt os</h3>

<ul className="list-unstyled">

<li>

<a className="text-gray-600 hover:text-gray-900">

<strong>E-mail: </strong>

<a>{import.meta.env.STOREEMAIL}</a>

</a>

</li>

<li>

<a className="text-gray-600 hover:text-gray-900">

<strong>TLF: </strong>

<a>{import.meta.env.STOREPHONE}</a>

<br></br>

<br></br>

</a>

</li>

<li>

<a className="text-gray-600 hover:text-gray-900">

<strong>Telefonåbningstider: </strong>

<br></br>

<a>Mandag - Fredag: 10:00 - 15:00</a>

</a>

</li>

</ul>

</Col>

{/\* Marketplace Column \*/}

<Col md={6} lg={3} className="mb-4">

<h3 className="text-lg font-semibold text-gray-900">Godt at vide</h3>

<ul className="list-unstyled">

<li className="mb-2">

<a href="/Privatpolitik" className="text-blue-500 hover:text-blue-600">

Information

</a>

</li>

<li className="mb-2">

<a

href="#"

className="text-gray-600 hover:text-gray-900"

onClick={(e) => {

e.preventDefault(); // Prevent default navigation if needed

const cookiesettings = localStorage.getItem("CookieSettings");

if (cookiesettings !== null) {

const preferences = JSON.parse(localStorage.getItem("CookieSettings") as string) as Settings

preferences.isvisible = true;

localStorage.setItem('CookieSettings', JSON.stringify(preferences));

window.location.href = "/"

}

}}

>

Cookie indstillinger

</a>

</li>

<li className="mb-2">

<a href="/Privatpolitik" className="text-blue-500 hover:text-blue-600">

Privatpolitik

</a>

</li>

</ul>

</Col>

{/\* Magazine Column \*/}

<Col md={6} lg={3} className="mb-4">

<h3 className="text-lg font-semibold text-gray-900">Kunderservice</h3>

<ul className="list-unstyled">

<li className="mb-2">

<a href="/Privatpolitik" className="text-blue-500 hover:text-blue-600">

Fragt - og levering

</a>

</li>

<li className="mb-2">

<a href="/Privatpolitik" className="text-blue-500 hover:text-blue-600">

Handelsbetingelser

</a>

</li>

<li className="mb-2">

<a href="/Privatpolitik" className="text-blue-500 hover:text-blue-600">

Privatpolitik

</a>

</li>

</ul>

</Col>

</Row>

{/\* Footer Bottom \*/}

<Row className="pt-3 border-top">

<Col className="d-flex justify-content-between flex-wrap">

<div className="mb-2">

<a href="/Privatpolitik" className="text-gray-600 hover:text-gray-900 me-3">

<strong>Privatpolitik</strong>

</a>

<strong>| </strong>

<a

href="#"

className="text-gray-600 hover:text-gray-900"

onClick={(e) => {

e.preventDefault(); // Prevent default navigation if needed

const cookiesettings = localStorage.getItem("CookieSettings");

if (cookiesettings !== null) {

const preferences = JSON.parse(localStorage.getItem("CookieSettings") as string) as Settings

preferences.isvisible = true;

localStorage.setItem('CookieSettings', JSON.stringify(preferences));

window.location.href = "/"

}

}}

>

<strong>Cookie indstillinger</strong>

<strong>| </strong>

<strong>CVR: {import.meta.env.STORECVR}</strong>

</a>

</div>

<div className="trustpilot-widget" data-locale="da-DK" data-template-id="56278e9abfbbba0bdcd568bc" data-businessunit-id="67961ff417619e19c8e61c8f" data-style-height="52px" data-style-width="100%">

<a href="https://dk.trustpilot.com/review/trumpbuybutik.dk" target="\_blank" rel="noopener">Trustpilot</a>

</div>

</Col>

</Row>

</Container>

</footer>

);

}

import React, { useEffect, useState } from 'react';

import { GetAllProducts, type Product } from '../../../FrontendRequests/Requests-Api/Product';

import { FetchAllCateGories, type Category } from '../../../FrontendRequests/Requests-Api/Category';

import CardProduct from '../products/cardProduct';

import { Card, ListGroup, Row, Col } from "react-bootstrap";

function ProductDisplay() {

const [GetProducts, SetProducts] = useState<Product[] | null>(null);

const [GetCategories, SetCategories] = useState<Category[] | null>(null);

const [selectedCategory, setSelectedCategory] = useState<number | null>(null);

const [searchQuery, setSearchQuery] = useState('');

const [selectedSort, setSelectedSort] = useState<string>('Pris: Lav til Høj');

function filterProductsWithQuantity(products: Product[]): Product[] {

return products.filter(product => {

const totalQuantity = product.productcolorsizes.reduce((sum, colorSize) => sum + colorSize.quantity, 0);

return totalQuantity > 0;

});

}

const SetupFilterByUrl = (GetCategories: Category[]) => {

// Extract categoryid from URL if present

const urlSearchParams = new URLSearchParams(window.location.search);

const params = Object.fromEntries(urlSearchParams.entries());

if (params.categoryid) {

const categoryId = Number(params.categoryid);

if (GetCategories) {

const categoryExists = GetCategories.some(category => category.categoryid === categoryId);

if (categoryExists) {

setSelectedCategory(categoryId);

const componentElement = document.getElementById("productsection");

if (componentElement) {

componentElement.scrollIntoView({

behavior: "smooth", // Optional: Smooth scroll animation

block: "center",

});

}

}

}

}

}

const filteredProducts = GetProducts?.filter((product) => {

const matchesSearch = product.name.toLowerCase().includes(searchQuery.toLowerCase());

const matchesCategory = selectedCategory

? product.categories.some((category) => category.categoryid === selectedCategory)

: true;

return matchesSearch && matchesCategory;

});

// Sort products based on selected sorting option

const sortedProducts = () => {

if (selectedSort === 'Pris: Lav til Høj') {

return filteredProducts?.sort((a, b) => a.salgpris\_ex\_moms - b.salgpris\_ex\_moms);

} else if (selectedSort === 'Pris: Høj til Lav') {

return filteredProducts?.sort((a, b) => b.salgpris\_ex\_moms - a.salgpris\_ex\_moms);

} else if (selectedSort === 'Navn') {

return filteredProducts?.sort((a, b) => a.name.localeCompare(b.name));

} else if (selectedSort === 'Producent') {

return filteredProducts?.sort((a, b) => a.manufacturer.localeCompare(b.manufacturer));

}

return filteredProducts;

};

const GetProcuts = async () => {

try {

const result = await GetAllProducts();

if (result.result) {

const products = filterProductsWithQuantity(result.data as Product[]);

SetProducts(products);

}

} catch (error) {

console.error('Error Getting products:', error);

}

};

// Fetch all categories from the API

const GetCategory = async () => {

try {

const result = await FetchAllCateGories();

if (result.result) {

SetCategories(result.data);

SetupFilterByUrl(result.data);

}

} catch (error) {

console.error('Error Getting Category:', error);

}

};

// UseEffect to load data on component mount

useEffect(() => {

GetProcuts();

GetCategory();

}, []); // Re-run effect if location.search changes or categories change

return (

<div className="container py-4">

<Row>

{/\* Sidebar \*/}

<Col md={2}>

<Card className="shadow-sm border-0 mt-6">

<Card.Body>

<h5 className="fw-bold mb-4">Kategorier</h5>

<ListGroup variant="flush">

{GetCategories?.map((category) => (

<ListGroup.Item

key={category.categoryid}

className={`d-flex justify-content-between align-items-center border-0 py-2 ${selectedCategory === category.categoryid ? 'bg-success text-white' : 'hoverable-category'}`}

style={{ cursor: 'pointer' }}

onClick={() => setSelectedCategory(category.categoryid === selectedCategory ? null : category.categoryid)}

>

{category.categoryname} <i className="mdi mdi-star ms-2"></i>

</ListGroup.Item>

))}

</ListGroup>

</Card.Body>

</Card>

</Col>

{/\* Main Content \*/}

<Col md={10}>

<div className="d-flex justify-content-between align-items-center mb-4">

<span className="text-muted">Viser {sortedProducts()?.length} Produkter</span>

<div className="d-flex align-items-center">

{/\* Sorting Dropdown \*/}

<li className="nav-item dropdown pe-2 d-flex align-items-center">

<a

href="#"

className="p-0 nav-link"

id="dropdownMenuButton"

data-bs-toggle="dropdown"

aria-expanded="false"

>

<i

className="mdi mdi-sort-variant"

style={{

fontSize: '24px',

color: 'currentColor',

marginBottom: '4px',

transition: 'color 0.3s ease',

}}

onMouseEnter={(e) => e.currentTarget.style.color = 'green'}

onMouseLeave={(e) => e.currentTarget.style.color = 'currentColor'}

></i>

</a>

<ul

className="px-2 py-3 dropdown-menu dropdown-menu-end me-sm-n4"

aria-labelledby="dropdownMenuButton"

>

<li className="mb-2">

<a

className="dropdown-item border-radius-md"

onClick={() => setSelectedSort('Navn')}

>

<div className="py-1 d-flex">

<svg

aria-hidden="true"

xmlns="http://www.w3.org/2000/svg"

width="24"

height="24"

fill="none"

viewBox="0 0 24 24"

onMouseEnter={(e) => e.currentTarget.style.color = 'green'}

onMouseLeave={(e) => e.currentTarget.style.color = 'currentColor'}

>

<path

stroke="currentColor"

stroke-linecap="round"

stroke-linejoin="round"

stroke-width="2"

d="M10 19H5a1 1 0 0 1-1-1v-1a3 3 0 0 1 3-3h2m10 1a3 3 0 0 1-3 3m3-3a3 3 0 0 0-3-3m3 3h1m-4 3a3 3 0 0 1-3-3m3 3v1m-3-4a3 3 0 0 1 3-3m-3 3h-1m4-3v-1m-2.121 1.879-.707-.707m5.656 5.656-.707-.707m-4.242 0-.707.707m5.656-5.656-.707.707M12 8a3 3 0 1 1-6 0 3 3 0 0 1 6 0Z"

/>

</svg>

<div className="d-flex flex-column justify-content-center">

<span className="font-weight-bold ps-2 fg-dark"

onMouseEnter={(e) => e.currentTarget.style.color = 'green'}

onMouseLeave={(e) => e.currentTarget.style.color = 'currentColor'}

>Navn</span>

</div>

</div>

</a>

</li>

<li className="mb-2">

<a

className="dropdown-item border-radius-md"

onClick={() => setSelectedSort('Producent')}

>

<div className="py-1 d-flex">

<svg

aria-hidden="true"

xmlns="http://www.w3.org/2000/svg"

width="24"

height="24"

fill="none"

viewBox="0 0 24 24"

onMouseEnter={(e) => e.currentTarget.style.color = 'green'}

onMouseLeave={(e) => e.currentTarget.style.color = 'currentColor'}

>

<path

stroke="currentColor"

stroke-linecap="round"

stroke-linejoin="round"

stroke-width="2"

d="m15 9-6 6m0-6 6 6m6-3a9 9 0 1 1-18 0 9 9 0 0 1 18 0Z"

/>

</svg>

<div className="d-flex flex-column justify-content-center">

<span className="font-weight-bold ps-2 fg-dark"

onMouseEnter={(e) => e.currentTarget.style.color = 'green'}

onMouseLeave={(e) => e.currentTarget.style.color = 'currentColor'}

>Producent</span>

</div>

</div>

</a>

</li>

<li className="mb-2">

<a

className="dropdown-item border-radius-md"

onClick={() => setSelectedSort('Pris: Lav til Høj')}

>

<div className="py-1 d-flex">

<svg

aria-hidden="true"

xmlns="http://www.w3.org/2000/svg"

width="24"

height="24"

fill="none"

viewBox="0 0 24 24"

onMouseEnter={(e) => e.currentTarget.style.color = 'green'}

onMouseLeave={(e) => e.currentTarget.style.color = 'currentColor'}

>

<path

stroke="currentColor"

stroke-linecap="round"

stroke-linejoin="round"

stroke-width="2"

d="M5 10l7-7m0 0l7 7m-7-7v18"

/>

</svg>

<div className="d-flex flex-column justify-content-center">

<span className="font-weight-bold ps-2 fg-dark"

onMouseEnter={(e) => e.currentTarget.style.color = 'green'}

onMouseLeave={(e) => e.currentTarget.style.color = 'currentColor'}

>Pris: Lav til Høj</span>

</div>

</div>

</a>

</li>

<li className="mb-2">

<a

className="dropdown-item border-radius-md"

onClick={() => setSelectedSort('Pris: Høj til Lav')}

>

<div className="py-1 d-flex">

<svg

aria-hidden="true"

xmlns="http://www.w3.org/2000/svg"

width="24"

height="24"

fill="none"

viewBox="0 0 24 24"

onMouseEnter={(e) => e.currentTarget.style.color = 'green'}

onMouseLeave={(e) => e.currentTarget.style.color = 'currentColor'}

>

<path

stroke="currentColor"

stroke-linecap="round"

stroke-linejoin="round"

stroke-width="2"

d="M19 14l-7 7m0 0l-7-7m7 7V3"

/>

</svg>

<div className="d-flex flex-column justify-content-center">

<span className="font-weight-bold ps-2 fg-dark"

onMouseEnter={(e) => e.currentTarget.style.color = 'green'}

onMouseLeave={(e) => e.currentTarget.style.color = 'currentColor'}

>Pris: Høj til Lav</span>

</div>

</div>

</a>

</li>

</ul>

</li>

{/\* Search Bar \*/}

<input

type="text"

className="form-control ms-2"

placeholder="Søg produkter..."

value={searchQuery}

onChange={(e) => setSearchQuery(e.target.value)}

/>

</div>

{/\* Sorting Dropdown end \*/}

</div>

{/\* Placeholder for product grid \*/}

<div className="border border-dashed p-4 text-center text-muted">

<div className="row">

{sortedProducts()?.map((product) => (

<div className="col-md-6 col-lg-3" key={product.productid}>

<CardProduct Product={product} />

</div>

))}

</div>

</div>

</Col>

</Row>

</div>

);

}

export default ProductDisplay;

import ShippingInfo from '../profile/shippingInfo';

import BillingInfo from '../profile/billingInfo';

import OrderSummary from '../cart/orderSummary';

import CheckoutSingleItem from '../checkout/checkoutSingleItem';

import React, {useEffect, useState } from 'react';

import { Button, Modal } from 'react-bootstrap';

import { useNotifications, type ExtendedBasketProduct } from '../NotifcationSystem/NotificationContext';

import { GetProduct } from '../../../FrontendRequests/Requests-Api/Product';

import { IsLoggedIn, fetchUserById } from '../../../FrontendRequests/Requests-Api/User';

import type { User } from '../../../FrontendRequests/Requests-Api/User';

import ContactInfo from '../profile/contactInfo';

import type { checkoutcontact, checkoutaddress, CheckoutDeliveryOption } from '../../../FrontendRequests/Interfaces/Checkout';

import DeliveryOptions from './DeliveryInfo';

import {UserRoles} from "../../../FrontendRequests/Enums/Usertypes"

import StripeForm from '../thirdparty/stripecheckout';

import type { stripecheckout } from '../../../FrontendRequests/Requests-Api/stripe';

import { getShippingPrice } from '../../../FrontendRequests/Enums/shippingprice';

export default function CheckoutSummary({

}) {

const { Basket, removeFromBasket } = useNotifications();

const [GetProducts, SetProducts] = useState<ExtendedBasketProduct[] | null>(null);

const [LoggedIn, setIsLoggedIn] = useState<boolean>(false);

const [sameAsShipping, setSameAsShipping] = useState(false);

const [User, SetUser] = useState<User>();

const [showWarningModal, setWarningModal] = useState(false);

const [showModal, setShowModal] = useState<boolean>(false);

const [selectedStripeCheckout, setselectedStripeCheckout] = useState<stripecheckout>();

const [handelsbetingelser, sethandelsbetingelser] = useState<boolean>(false);

const handleClose = () => setShowModal(false);

const [contactInfo, setContactInfo] = useState<checkoutcontact>({

result: false,

email: "",

firstname: "",

lastname: "",

});

const [addressinfo, setAddressInfo] = useState<checkoutaddress>({

result: false,

address: "",

phonenumber: 0,

city: "",

country: "",

postalcode: 0,

});

const [DeliveryOption, setDeliveryOption] = useState<CheckoutDeliveryOption>({

IsParcelShop: false,

IsHomeDelivery: false,

price: 0

});

const checkLoginStatus = async () => {

try {

const result = await IsLoggedIn();

setIsLoggedIn(result.result);

if (result.result) {

const userresult = await fetchUserById(result.data.user.userid)

SetUser(userresult.data)

}

} catch (error) {

console.error('Error checking login status:', error);

setIsLoggedIn(false);

}

};

const GetProductsByIds = async () => {

try {

const productPromises = Basket.map(async (product, index) => {

const response = await GetProduct(product.ProductId.toString()); // Fetch product by ID

if (response.result) { // Check if response.result is truthy

return {

...response.data, // Spread the rest of the product data

productcolorsizes: [product.ProductCustoms],

quantity: product.quantity,

};

} else {

removeFromBasket(index); // Remove the product from the basket

console.log(`Product at index ${index} removed from basket`);

return null; // Return null for invalid products

}

});

// Wait for all promises to resolve

const updatedProducts = await Promise.all(productPromises);

// Filter out any null values and update the products

const validProducts = updatedProducts.filter(product => product !== null);

SetProducts(validProducts); // Assuming SetProducts updates the state

} catch (error) {

console.error('Error fetching products:', error);

}

};

// Handle button click

const handleProceedToPaymentClick = () => {

if (GetProducts && GetProducts?.length > 0) {

if (contactInfo.result && addressinfo.result) {

if (handelsbetingelser) {

if (DeliveryOption.IsHomeDelivery) {

const userId = LoggedIn ? User?.userid : UserRoles.DEFAULT;

const homeAddress = `${User?.user\_address}, ${User?.address\_postalcode}, ${User?.address\_city}, ${User?.country\_name}`;

setselectedStripeCheckout({

userId: userId as number,

parceladdress: homeAddress,

isHomeDelivery: true,

products: GetProducts,

contactInfo: contactInfo,

addressinfo: addressinfo,

deliveryAddress: DeliveryOption

})

setShowModal(true);

} else if (DeliveryOption.IsParcelShop) {

const userId = LoggedIn ? User?.userid : UserRoles.DEFAULT;

const parcelShopAddress = `${DeliveryOption.ParcelShop?.name}, ${DeliveryOption.ParcelShop?.street}, ${DeliveryOption.ParcelShop?.houseNumber}, ${DeliveryOption.ParcelShop?.city}, ${DeliveryOption.ParcelShop?.zipCode}`;

setselectedStripeCheckout({

userId: userId as number,

parceladdress: parcelShopAddress,

isHomeDelivery: true,

products: GetProducts,

contactInfo: contactInfo,

addressinfo: addressinfo,

deliveryAddress: DeliveryOption

})

setShowModal(true);

}

} else {

}

}

} else {

setWarningModal(true);

}

};

let subtotal = 0;

GetProducts?.map(product =>

subtotal += parseFloat((Number(product.salgpris\_ex\_moms) + Number(product.udgående\_moms)).toString()) \* product.quantity

);

let moms = 0;

GetProducts?.map(product =>

moms += parseFloat((Number(product.udgående\_moms)).toString()) \* product.quantity

);

useEffect(() => {

checkLoginStatus();

GetProductsByIds();

}, []);

return (

<>

<section>

<div className="row">

<div className="col-12 col-lg-6 p-3 p-md-5 bg-gray-100">

<h5 className="mb-4">Kontakt information</h5>

<ContactInfo IsLoggedIn={LoggedIn} setContactInfo={setContactInfo} contactInfo={contactInfo as checkoutcontact} User={User as User}/>

<h5 className="mt-5 mb-4">Adresse</h5>

<ShippingInfo

sameAsShipping={sameAsShipping}

IsLoggedIn={LoggedIn}

setAddressInfo={setAddressInfo}

addressinfo={addressinfo}

User={User as User}

/>

<h5 className="mt-5 mb-4">Levering</h5>

<DeliveryOptions SetDeliveryOption={setDeliveryOption} price={getShippingPrice(Basket.length)}/>

{LoggedIn && (

<>

<h5 className="mt-5 mb-4">Faktureringsadresse</h5>

<BillingInfo

sameAsShipping={sameAsShipping}

setSameAsShipping={setSameAsShipping}

/>

</>

)}

<div>

<h5 className="mt-5 mb-4">Handelsbetingelser</h5>

<div className="form-check">

<input

className="form-check-input"

type="checkbox"

checked={handelsbetingelser}

onChange={() => sethandelsbetingelser(!handelsbetingelser)}

id="handelsbetingelserCheckbox"

/>

<label htmlFor="handelsbetingelserCheckbox" className="custom-control-label">

<strong style={{ textDecoration: 'underline' }}>

<a href="/Privatpolitik" style={{ textDecoration: 'none', color: 'inherit' }}>

Accepter handelsbetingelser

</a>

</strong>

</label>

{!handelsbetingelser && (

<div style={{ color: 'red', marginTop: '8px' }}>

Venligst accepter handelsbetingelser

</div>

)}

</div>

</div>

<hr className="dark horizontal"/>

<button className="btn btn-dark float-end mt-2 mb-0" onClick={handleProceedToPaymentClick}>

<svg className="me-1" width="14" height="14" viewBox="0 0 14 14" fill="none" xmlns="http://www.w3.org/2000/svg">

<path d="M2.80039 2.80005C2.02719 2.80005 1.40039 3.42685 1.40039 4.20005V4.90005H12.6004V4.20005C12.6004 3.42685 11.9736 2.80005 11.2004 2.80005H2.80039Z" fill="white"/>

<path fill-rule="evenodd" clip-rule="evenodd" d="M12.6004 6.30005H1.40039V9.80005C1.40039 10.5733 2.02719 11.2 2.80039 11.2H11.2004C11.9736 11.2 12.6004 10.5733 12.6004 9.80005V6.30005ZM2.80039 9.10005C2.80039 8.71344 3.11379 8.40005 3.50039 8.40005H4.20039C4.58699 8.40005 4.90039 8.71344 4.90039 9.10005C4.90039 9.48666 4.58699 9.80005 4.20039 9.80005H3.50039C3.11379 9.80005 2.80039 9.48666 2.80039 9.10005ZM6.30039 8.40005C5.91379 8.40005 5.60039 8.71344 5.60039 9.10005C5.60039 9.48666 5.91379 9.80005 6.30039 9.80005H7.00039C7.387 9.80005 7.70039 9.48666 7.70039 9.10005C7.70039 8.71344 7.387 8.40005 7.00039 8.40005H6.30039Z" fill="white"/>

</svg>

Fortsæt til betaling

</button>

</div>

<div className="col-12 col-lg-6 p-3 p-md-5 bg-dark bg-gradient rounded-end">

<p className="text-white opacity-6 mb-0 text-end">Pris</p>

<h3 className="text-white mb-4 text-end">{subtotal.toLocaleString()} kr.</h3>

{GetProducts?.map((product, i) =>

<CheckoutSingleItem

Product={product}

/>

)}

<OrderSummary subtotal={subtotal} shippingprice={getShippingPrice(GetProducts?.length)} moms={moms} />

</div>

</div>

{/\* Modal for stripe \*/}

<Modal show={showModal} onHide={handleClose}>

<Modal.Header closeButton>

<div className="w-100 text-center">

<Modal.Title>Powered by Stripe</Modal.Title>

</div>

</Modal.Header>

<Modal.Body>

<StripeForm stripecheckout={selectedStripeCheckout as stripecheckout} />

</Modal.Body>

<Modal.Footer>

<Button variant="secondary" onClick={handleClose}>

Close

</Button>

</Modal.Footer>

</Modal>

{/\* Modal for Warning \*/}

<Modal show={showWarningModal} size="lg" centered>

<Modal.Header closeButton className="text-white" style={{ backgroundColor: "red" }}>

<Modal.Title className="d-flex flex-column">

<div className="d-flex align-items-center">

<i className="mdi mdi-delete-empty-outline me-2" style={{ fontSize: "1.5rem" }}></i>

Basket is empty

</div>

<div className="mt-2 fw-bold" style={{ fontSize: "0.9rem" }}>

Please add products in basket to proceed

</div>

</Modal.Title>

</Modal.Header>

<Modal.Footer className="d-flex justify-content-center">

<button className="btn btn-success me-2" onClick={() => window.location.href = '/'}>Refresh</button>

</Modal.Footer>

</Modal>

</section>

</>

);

};

import { useEffect } from 'react';

import {AddVisitor} from "../../../FrontendRequests/Requests-Api/visitors"

const PageVisitTracker = () => {

useEffect(() => {

if (!sessionStorage.getItem('visited')) {

// Optionally, send this information to a server or logging service

AddVisitor().then((result) => {

})

sessionStorage.setItem('visited', 'true');

}

}, []);

return null;

};

export default PageVisitTracker;

import React, { useState, useEffect } from "react";

import ReactDOM from "react-dom";

import { Tabs, Tab, Form, Button } from 'react-bootstrap';

export interface Settings {

isvisible: boolean;

CookieSettings: CookieSettings

}

interface CookieSettings {

Nødvendig: boolean;

Præferencer: boolean;

Statistik: boolean;

Markedsføring: boolean;

}

const CookieBannerModal = () => {

const [isVisible, setIsVisible] = useState(true);

const [activeTab, setActiveTab] = useState('samtykke');

const [settings, setSettings] = useState<CookieSettings>({

Nødvendig: true,

Præferencer: false,

Statistik: false,

Markedsføring: false,

});

const [loading, setLoading] = useState(true);

const handleAllowAll = () => {

const preferences : Settings = {

isvisible : false,

CookieSettings: {

Nødvendig: true,

Præferencer: true,

Statistik: true,

Markedsføring: true,

} as CookieSettings

}

localStorage.setItem('CookieSettings', JSON.stringify(preferences));

setIsVisible(preferences.isvisible);

};

const handleAllowSelected = () => {

const preferences : Settings = {

isvisible : false,

CookieSettings: settings

}

localStorage.setItem('CookieSettings', JSON.stringify(preferences));

setIsVisible(preferences.isvisible);

};

const handleReject = () => {

const preferences : Settings = {

isvisible : false,

CookieSettings: {

Nødvendig: true,

Præferencer: false,

Statistik: false,

Markedsføring: false,

} as CookieSettings

}

localStorage.setItem('CookieSettings', JSON.stringify(preferences));

setIsVisible(preferences.isvisible);

};

useEffect(() => {

const cookiesettings = localStorage.getItem("CookieSettings");

if (cookiesettings !== null) {

const preferences = JSON.parse(localStorage.getItem("CookieSettings") as string) as Settings

if (preferences.isvisible) {

setSettings(preferences.CookieSettings)

setIsVisible(preferences.isvisible)

} else {

setSettings(preferences.CookieSettings)

setIsVisible(preferences.isvisible)

}

} else {

}

setLoading(false);

}, []);

if (loading) return null;

if (!isVisible) return null;

return ReactDOM.createPortal(

<div className="fixed-top fixed-bottom fixed-left fixed-right bg-opacity-50 d-flex align-items-center justify-content-center">

<div className="bg-white rounded-lg shadow-lg" style={{ maxWidth: '800px', width: '100%' }}>

<div className="d-flex justify-content-between align-items-center p-3 border-bottom">

<div className="fw-bold fs-5">{import.meta.env.STORENAME}</div>

<div className="text-muted small">Powered by {import.meta.env.STORENAME}</div>

</div>

<Tabs

activeKey={activeTab}

onSelect={(k) => setActiveTab(k || 'samtykke')}

className="mb-3"

>

<Tab eventKey="samtykke" title="Samtykke">

<div className="p-4">

<h2 className="fs-4 fw-semibold mb-3">Denne hjemmeside bruger cookies</h2>

<p className="text-muted small mb-4">

Vi ønsker dit samtykke til at anvende cookies og indsamle persondata om IP-adresse, ID og din browser til statistik og marketingformål. Disse oplysninger videregives til vores samarbejdspartnere, der opbevarer og tilgår oplysninger på din enhed for at vise dig målrettede annoncer, levere tilpasset indhold, foretage annonce- og indholdsmåling, lave produktudvikling og opnå målgruppeindsigt. Se mere information under indstillinger og i vores persondatapolitik.

</p>

<h3 className="fs-6 text-muted mb-2">Du kan altid:</h3>

<p className="text-muted small mb-4">Ændre dine valg i bunden af hjemmesiden ved at klikke på <strong>Cookie indstillinger</strong></p>

<div className="row mb-4">

{['Nødvendig', 'Præferencer', 'Statistik', 'Markedsføring'].map((label, index) => (

<div key={label} className="col-3 text-center">

<style>

{`

.custom-switch .form-check-input:checked {

background-color: ${import.meta.env.LIGHT\_PURPLE};

border-color: ${import.meta.env.LIGHT\_PURPLE};

}

`}

</style>

<Form.Check

type="switch"

id={`cookie-switch-${index}`}

label={label}

checked={settings[label as keyof CookieSettings]}

onChange={(e) => setSettings(prev => ({ ...prev, [label]: e.target.checked }))}

disabled={label === 'Nødvendig'}

className="custom-switch"

/>

</div>

))}

</div>

<div className="d-flex gap-2">

<Button variant="outline-secondary" onClick={handleReject} className="flex-grow-1">

Afvis

</Button>

<Button variant="outline-primary" onClick={handleAllowSelected} className="flex-grow-1">

Tillad valgte

</Button>

<Button variant="primary" onClick={handleAllowAll} className="flex-grow-1">

Tillad alle

</Button>

</div>

</div>

</Tab>

<Tab eventKey="Cookiepolitik" title="Cookie-Politik">

<div className="p-4">

<h3 className="fs-6 text-muted mb-2">Nødvendig</h3>

<p className="text-muted small mb-4">Sikrer webshoppen fungerer korrekt, fx ved login og betaling</p>

<h3 className="fs-6 text-muted mb-2">Præferencer</h3>

<p className="text-muted small mb-4">Husker dine valg, fx sprog og valutaindstillinger</p>

<h3 className="fs-6 text-muted mb-2">Statistik</h3>

<p className="text-muted small mb-4">Indsamler anonym data til at forbedre brugeroplevelsen</p>

<h3 className="fs-6 text-muted mb-2">Markedsføring</h3>

<p className="text-muted small mb-4">Bruges til målrettet reklame og tilbud</p>

</div>

</Tab>

<Tab eventKey="Privatpolitik" title="Privat-Politik">

<div className="p-4">

<p className="text-muted small mb-4">Du kan altid læse om vores privatpolitik ved at klikke på: </p>

<h3 className="fs-6 text-muted mb-2">

<a href="/Privatpolitik" className="text-muted text-decoration-none">Privatpolitik</a>

</h3>

</div>

</Tab>

</Tabs>

</div>

</div>,

document.body

);

};

export default CookieBannerModal;

import React from 'react';

import { Folders } from '../../../FrontendRequests/Enums/Images';

interface Props {

bannernumber: number;

}

export default function PromoSectionLarge({

bannernumber,

}: Props) {

return (

<section>

<a style={{ textDecoration: 'none' }}>

<div

className="promo-section"

style={{

backgroundImage: `url(https://res.cloudinary.com/${import.meta.env.CLOUDINARY\_CLOUD\_NAME}/image/upload/v${new Date().getTime()}/${Folders.MEDIA}/banner${bannernumber}.jpg)`,

backgroundSize: 'cover', // Ensure the image covers the div

backgroundPosition: 'top',

height: '360px', // Fixed height

borderRadius: '1rem',

cursor: 'pointer', // Indicate the container is clickable

}}

></div>

</a>

</section>

);

}

import React, { useEffect, useState } from 'react';

import { FetchAllCateGories } from '../../../FrontendRequests/Requests-Api/Category';

import type { Category } from '../../../FrontendRequests/Requests-Api/Category';

import { Card, Row, Col, Container } from 'react-bootstrap';

export default function CardCategory({ maxItems = 4 }) {

const [categories, setCategories] = useState<Category[] | null>(null);

const uniqueClass = `category-card-${Math.random().toString(36).substring(2, 9)}`; // Unique class name

const fetchCategories = async () => {

try {

const result = await FetchAllCateGories();

if (result.result) {

const shuffledCategories: Category[] = result.data.sort(() => Math.random() - 0.5);

setCategories(shuffledCategories.slice(0, maxItems));

}

} catch (error) {

console.error('Error Fetching Categories:', error);

}

};

useEffect(() => {

fetchCategories();

}, [maxItems]);

return (

<Container>

{/\* Scoped Styles \*/}

<style>

{`

.${uniqueClass} {

position: relative;

overflow: hidden;

border-radius: 8px;

height: 180px;

}

.${uniqueClass} .category-image {

width: 100%;

height: 100%;

background-size: cover;

background-position: center;

transition: transform 0.3s ease-in-out;

}

.${uniqueClass}:hover .category-image {

transform: scale(1.1);

}

.${uniqueClass} .card-body {

position: absolute;

bottom: 0;

width: 100%;

background: rgba(0, 0, 0, 0.6);

padding: 8px;

}

`}

</style>

<Row className="justify-content-center mb-4 mt-4">

{categories?.map((category, index) => (

<Col xs={12} sm={6} md={4} lg={3} key={index} className="mb-4">

<a href={`/?categoryid=${category.categoryid}`} className="text-decoration-none">

<Card className={`${uniqueClass} border-0`}>

<div

className="category-image"

style={{

backgroundImage: `url(${category.categoryimage})`,

}}

></div>

<Card.Body className="text-center p-2">

<Card.Title as="p" className="text-white fw-bold mb-1 small">

{category.categoryname}

</Card.Title>

<a href={`/?categoryid=${category.categoryid}`} className="text-white text-sm fw-semibold">

Flere produkter &#62;

</a>

</Card.Body>

</Card>

</a>

</Col>

))}

</Row>

</Container>

);

}

import React from "react";

interface Props {

// order: string;

}

export default function incentiveCols({

}: Props) {

return (

<>

<div className="container">

<div className="row mt-5">

<div className="col-12 col-md-4">

<div className="card shadow-xs border">

<div className="card-body p-3 d-flex align-items-center justify-content-center">

<div className="icon icon-shape bg-dark text-white text-center border-radius-sm d-flex align-items-center justify-content-center">

<svg width="24" height="24" viewBox="0 0 24 24" fill="none" xmlns="http://www.w3.org/2000/svg">

<path d="M9.6 19.8C9.6 20.7941 8.79412 21.6 7.8 21.6C6.80588 21.6 6 20.7941 6 19.8C6 18.8059 6.80588 18 7.8 18C8.79412 18 9.6 18.8059 9.6 19.8Z" fill="white"/>

<path d="M18.0004 19.8C18.0004 20.7941 17.1945 21.6 16.2004 21.6C15.2063 21.6 14.4004 20.7941 14.4004 19.8C14.4004 18.8059 15.2063 18 16.2004 18C17.1945 18 18.0004 18.8059 18.0004 19.8Z" fill="#F9FAFB"/>

<path d="M3.60039 4.80005C2.93765 4.80005 2.40039 5.33731 2.40039 6.00005V18C2.40039 18.6628 2.93765 19.2 3.60039 19.2H4.8604C5.13836 17.8307 6.34901 16.8 7.80039 16.8C9.25177 16.8 10.4624 17.8307 10.7404 19.2H12.0004C12.6631 19.2 13.2004 18.6628 13.2004 18V6.00005C13.2004 5.33731 12.6631 4.80005 12.0004 4.80005H3.60039Z" fill="#F9FAFB"/>

<path d="M16.7996 8.40002C16.1368 8.40002 15.5996 8.93729 15.5996 9.60002V16.86C15.7935 16.8207 15.9942 16.8 16.1996 16.8C17.651 16.8 18.8617 17.8307 19.1396 19.2H20.3996C21.0624 19.2 21.5996 18.6628 21.5996 18V12C21.5996 11.6818 21.4731 11.3765 21.2481 11.1515L18.8481 8.75149C18.6231 8.52646 18.3178 8.40002 17.9996 8.40002H16.7996Z" fill="#F9FAFB"/>

</svg>

</div>

<div className="ms-3">

<h5 className="mb-0">Ekspresforsendelse</h5>

<p className="mb-0 opacity-8">Har du brug for det hurtigt? Bare rolig!</p>

</div>

</div>

</div>

</div>

<div className="col-12 col-md-4 my-4 my-md-0">

<div className="card shadow-xs border">

<div className="card-body p-3 d-flex align-items-center justify-content-center">

<div className="icon icon-shape bg-dark text-white text-center border-radius-sm d-flex align-items-center justify-content-center">

<svg width="25" height="24" viewBox="0 0 25 24" fill="none" xmlns="http://www.w3.org/2000/svg">

<path fill-rule="evenodd" clip-rule="evenodd" d="M5.13359 2.40002C5.79633 2.40002 6.33359 2.93729 6.33359 3.60002V6.12152C7.8577 4.56589 9.98222 3.60002 12.3336 3.60002C15.9929 3.60002 19.1028 5.93921 20.2553 9.20012C20.4762 9.82499 20.1487 10.5106 19.5238 10.7314C18.8989 10.9523 18.2134 10.6248 17.9924 9.99992C17.1683 7.66792 14.9442 6.00002 12.3336 6.00002C10.3717 6.00002 8.62804 6.94205 7.53281 8.40002H11.1336C11.7963 8.40002 12.3336 8.93729 12.3336 9.60002C12.3336 10.2628 11.7963 10.8 11.1336 10.8H5.13359C4.47086 10.8 3.93359 10.2628 3.93359 9.60002V3.60002C3.93359 2.93729 4.47086 2.40002 5.13359 2.40002ZM5.1434 13.2687C5.76825 13.0477 6.45383 13.3752 6.67469 14.0002C7.49894 16.3321 9.72303 18 12.3336 18C14.2955 18 16.0392 17.058 17.1343 15.6H13.5336C12.8708 15.6 12.3336 15.0628 12.3336 14.4C12.3336 13.7373 12.8708 13.2 13.5336 13.2H19.5336C19.8518 13.2 20.1571 13.3265 20.3821 13.5515C20.6071 13.7765 20.7336 14.0818 20.7336 14.4V20.4C20.7336 21.0628 20.1964 21.6 19.5336 21.6C18.8708 21.6 18.3336 21.0628 18.3336 20.4V17.8785C16.8095 19.4341 14.685 20.4 12.3336 20.4C8.6743 20.4 5.56445 18.0609 4.41188 14.7999C4.19102 14.175 4.51853 13.4895 5.1434 13.2687Z" fill="white"/>

</svg>

</div>

<div className="ms-3">

<h5 className="mb-0">Returret</h5>

<p className="mb-0 opacity-8">14 dags returet</p>

</div>

</div>

</div>

</div>

<div className="col-12 col-md-4">

<div className="card shadow-xs border">

<div className="card-body p-3 d-flex align-items-center justify-content-center">

<div className="icon icon-shape bg-dark text-white text-center border-radius-sm d-flex align-items-center justify-content-center">

<svg width="25" height="24" viewBox="0 0 25 24" fill="none" xmlns="http://www.w3.org/2000/svg">

<path fill-rule="evenodd" clip-rule="evenodd" d="M3.26574 5.99871C6.87565 5.94973 10.1647 4.57242 12.6664 2.33337C15.168 4.57242 18.4571 5.94973 22.0671 5.99871C22.1981 6.77964 22.2664 7.58187 22.2664 8.40006C22.2664 14.6699 18.2594 20.0038 12.6664 21.9807C7.07348 20.0038 3.06641 14.6699 3.06641 8.40006C3.06641 7.58187 3.13464 6.77964 3.26574 5.99871ZM17.1149 10.4486C17.5835 9.97992 17.5835 9.22013 17.1149 8.75149C16.6463 8.28287 15.8865 8.28287 15.4179 8.75149L11.4664 12.703L9.91494 11.1515C9.4463 10.6829 8.68651 10.6829 8.21787 11.1515C7.74925 11.6201 7.74925 12.3799 8.21787 12.8485L10.6179 15.2485C11.0865 15.7171 11.8463 15.7171 12.3149 15.2485L17.1149 10.4486Z" fill="white"/>

</svg>

</div>

<div className="ms-3">

<h5 className="mb-0">Reklamation</h5>

<p className="mb-0 opacity-8">2 års reklamation</p>

</div>

</div>

</div>

</div>

</div>

</div>

</>

);

};

import React, { useEffect, useState } from 'react';

import { GetAllProducts, type Product } from '../../../FrontendRequests/Requests-Api/Product';

import { FetchAllCateGories, type Category } from '../../../FrontendRequests/Requests-Api/Category';

import CardProduct from '../products/cardProduct';

import { Card, ListGroup, Row, Col } from "react-bootstrap";

function ProductDisplay() {

const [GetProducts, SetProducts] = useState<Product[] | null>(null);

const [GetCategories, SetCategories] = useState<Category[] | null>(null);

const [selectedCategory, setSelectedCategory] = useState<number | null>(null);

const [searchQuery, setSearchQuery] = useState('');

const [selectedSort, setSelectedSort] = useState<string>('Pris: Lav til Høj');

function filterProductsWithQuantity(products: Product[]): Product[] {

return products.filter(product => {

const totalQuantity = product.productcolorsizes.reduce((sum, colorSize) => sum + colorSize.quantity, 0);

return totalQuantity > 0;

});

}

const SetupFilterByUrl = (GetCategories: Category[]) => {

// Extract categoryid from URL if present

const urlSearchParams = new URLSearchParams(window.location.search);

const params = Object.fromEntries(urlSearchParams.entries());

if (params.categoryid) {

const categoryId = Number(params.categoryid);

if (GetCategories) {

const categoryExists = GetCategories.some(category => category.categoryid === categoryId);

if (categoryExists) {

setSelectedCategory(categoryId);

const componentElement = document.getElementById("productsection");

if (componentElement) {

componentElement.scrollIntoView({

behavior: "smooth", // Optional: Smooth scroll animation

block: "center",

});

}

}

}

}

}

const filteredProducts = GetProducts?.filter((product) => {

const matchesSearch = product.name.toLowerCase().includes(searchQuery.toLowerCase());

const matchesCategory = selectedCategory

? product.categories.some((category) => category.categoryid === selectedCategory)

: true;

return matchesSearch && matchesCategory;

});

// Sort products based on selected sorting option

const sortedProducts = () => {

if (selectedSort === 'Pris: Lav til Høj') {

return filteredProducts?.sort((a, b) => a.salgpris\_ex\_moms - b.salgpris\_ex\_moms);

} else if (selectedSort === 'Pris: Høj til Lav') {

return filteredProducts?.sort((a, b) => b.salgpris\_ex\_moms - a.salgpris\_ex\_moms);

} else if (selectedSort === 'Navn') {

return filteredProducts?.sort((a, b) => a.name.localeCompare(b.name));

} else if (selectedSort === 'Producent') {

return filteredProducts?.sort((a, b) => a.manufacturer.localeCompare(b.manufacturer));

}

return filteredProducts;

};

const GetProcuts = async () => {

try {

const result = await GetAllProducts();

if (result.result) {

const products = filterProductsWithQuantity(result.data as Product[]);

SetProducts(products);

}

} catch (error) {

console.error('Error Getting products:', error);

}

};

// Fetch all categories from the API

const GetCategory = async () => {

try {

const result = await FetchAllCateGories();

if (result.result) {

SetCategories(result.data);

SetupFilterByUrl(result.data);

}

} catch (error) {

console.error('Error Getting Category:', error);

}

};

// UseEffect to load data on component mount

useEffect(() => {

GetProcuts();

GetCategory();

}, []); // Re-run effect if location.search changes or categories change

return (

<div className="container py-4">

<Row>

{/\* Sidebar \*/}

<Col md={2}>

<Card className="shadow-sm border-0 mt-6">

<Card.Body>

<h5 className="fw-bold mb-4">Kategorier</h5>

<ListGroup variant="flush">

{GetCategories?.map((category) => (

<ListGroup.Item

key={category.categoryid}

className={`d-flex justify-content-between align-items-center border-0 py-2 ${selectedCategory === category.categoryid ? 'bg-success text-white' : 'hoverable-category'}`}

style={{ cursor: 'pointer' }}

onClick={() => setSelectedCategory(category.categoryid === selectedCategory ? null : category.categoryid)}

>

{category.categoryname} <i className="mdi mdi-star ms-2"></i>

</ListGroup.Item>

))}

</ListGroup>

</Card.Body>

</Card>

</Col>

{/\* Main Content \*/}

<Col md={10}>

<div className="d-flex justify-content-between align-items-center mb-4">

<span className="text-muted">Viser {sortedProducts()?.length} Produkter</span>

<div className="d-flex align-items-center">

{/\* Sorting Dropdown \*/}

<li className="nav-item dropdown pe-2 d-flex align-items-center">

<a

href="#"

className="p-0 nav-link"

id="dropdownMenuButton"

data-bs-toggle="dropdown"

aria-expanded="false"

>

<i

className="mdi mdi-sort-variant"

style={{

fontSize: '24px',

color: 'currentColor',

marginBottom: '4px',

transition: 'color 0.3s ease',

}}

onMouseEnter={(e) => e.currentTarget.style.color = 'green'}

onMouseLeave={(e) => e.currentTarget.style.color = 'currentColor'}

></i>

</a>

<ul

className="px-2 py-3 dropdown-menu dropdown-menu-end me-sm-n4"

aria-labelledby="dropdownMenuButton"

>

<li className="mb-2">

<a

className="dropdown-item border-radius-md"

onClick={() => setSelectedSort('Navn')}

>

<div className="py-1 d-flex">

<svg

aria-hidden="true"

xmlns="http://www.w3.org/2000/svg"

width="24"

height="24"

fill="none"

viewBox="0 0 24 24"

onMouseEnter={(e) => e.currentTarget.style.color = 'green'}

onMouseLeave={(e) => e.currentTarget.style.color = 'currentColor'}

>

<path

stroke="currentColor"

stroke-linecap="round"

stroke-linejoin="round"

stroke-width="2"

d="M10 19H5a1 1 0 0 1-1-1v-1a3 3 0 0 1 3-3h2m10 1a3 3 0 0 1-3 3m3-3a3 3 0 0 0-3-3m3 3h1m-4 3a3 3 0 0 1-3-3m3 3v1m-3-4a3 3 0 0 1 3-3m-3 3h-1m4-3v-1m-2.121 1.879-.707-.707m5.656 5.656-.707-.707m-4.242 0-.707.707m5.656-5.656-.707.707M12 8a3 3 0 1 1-6 0 3 3 0 0 1 6 0Z"

/>

</svg>

<div className="d-flex flex-column justify-content-center">

<span className="font-weight-bold ps-2 fg-dark"

onMouseEnter={(e) => e.currentTarget.style.color = 'green'}

onMouseLeave={(e) => e.currentTarget.style.color = 'currentColor'}

>Navn</span>

</div>

</div>

</a>

</li>

<li className="mb-2">

<a

className="dropdown-item border-radius-md"

onClick={() => setSelectedSort('Producent')}

>

<div className="py-1 d-flex">

<svg

aria-hidden="true"

xmlns="http://www.w3.org/2000/svg"

width="24"

height="24"

fill="none"

viewBox="0 0 24 24"

onMouseEnter={(e) => e.currentTarget.style.color = 'green'}

onMouseLeave={(e) => e.currentTarget.style.color = 'currentColor'}

>

<path

stroke="currentColor"

stroke-linecap="round"

stroke-linejoin="round"

stroke-width="2"

d="m15 9-6 6m0-6 6 6m6-3a9 9 0 1 1-18 0 9 9 0 0 1 18 0Z"

/>

</svg>

<div className="d-flex flex-column justify-content-center">

<span className="font-weight-bold ps-2 fg-dark"

onMouseEnter={(e) => e.currentTarget.style.color = 'green'}

onMouseLeave={(e) => e.currentTarget.style.color = 'currentColor'}

>Producent</span>

</div>

</div>

</a>

</li>

<li className="mb-2">

<a

className="dropdown-item border-radius-md"

onClick={() => setSelectedSort('Pris: Lav til Høj')}

>

<div className="py-1 d-flex">

<svg

aria-hidden="true"

xmlns="http://www.w3.org/2000/svg"

width="24"

height="24"

fill="none"

viewBox="0 0 24 24"

onMouseEnter={(e) => e.currentTarget.style.color = 'green'}

onMouseLeave={(e) => e.currentTarget.style.color = 'currentColor'}

>

<path

stroke="currentColor"

stroke-linecap="round"

stroke-linejoin="round"

stroke-width="2"

d="M5 10l7-7m0 0l7 7m-7-7v18"

/>

</svg>

<div className="d-flex flex-column justify-content-center">

<span className="font-weight-bold ps-2 fg-dark"

onMouseEnter={(e) => e.currentTarget.style.color = 'green'}

onMouseLeave={(e) => e.currentTarget.style.color = 'currentColor'}

>Pris: Lav til Høj</span>

</div>

</div>

</a>

</li>

<li className="mb-2">

<a

className="dropdown-item border-radius-md"

onClick={() => setSelectedSort('Pris: Høj til Lav')}

>

<div className="py-1 d-flex">

<svg

aria-hidden="true"

xmlns="http://www.w3.org/2000/svg"

width="24"

height="24"

fill="none"

viewBox="0 0 24 24"

onMouseEnter={(e) => e.currentTarget.style.color = 'green'}

onMouseLeave={(e) => e.currentTarget.style.color = 'currentColor'}

>

<path

stroke="currentColor"

stroke-linecap="round"

stroke-linejoin="round"

stroke-width="2"

d="M19 14l-7 7m0 0l-7-7m7 7V3"

/>

</svg>

<div className="d-flex flex-column justify-content-center">

<span className="font-weight-bold ps-2 fg-dark"

onMouseEnter={(e) => e.currentTarget.style.color = 'green'}

onMouseLeave={(e) => e.currentTarget.style.color = 'currentColor'}

>Pris: Høj til Lav</span>

</div>

</div>

</a>

</li>

</ul>

</li>

{/\* Search Bar \*/}

<input

type="text"

className="form-control ms-2"

placeholder="Søg produkter..."

value={searchQuery}

onChange={(e) => setSearchQuery(e.target.value)}

/>

</div>

{/\* Sorting Dropdown end \*/}

</div>

{/\* Placeholder for product grid \*/}

<div className="border border-dashed p-4 text-center text-muted">

<div className="row">

{sortedProducts()?.map((product) => (

<div className="col-md-6 col-lg-3" key={product.productid}>

<CardProduct Product={product} />

</div>

))}

</div>

</div>

</Col>

</Row>

</div>

);

}

export default ProductDisplay;

import React, { useEffect, useState } from 'react';

import { FetchAllCateGories } from '../../../FrontendRequests/Requests-Api/Category';

import type { Category } from '../../../FrontendRequests/Requests-Api/Category';

export default function Categories() {

const [GetCategories, SetCategories] = useState<Category[] | null>(null);

const GetCategory = async () => {

try {

const result = await FetchAllCateGories();

if (result.result) {

const categoryarray : Category[] = result.data.sort(() => Math.random() - 0.5)

SetCategories(categoryarray);

}

} catch (error) {

console.error('Error Getting Category:', error);

}

};

useEffect(() => {

GetCategory();

}, []);

const classBody = 'text-center w-100 pt-8';

return (

<div className="row mb-5 mt-5">

{GetCategories !== null &&

GetCategories.map((category, index) => (

<div className="col-md-6 col-lg-3 mb-5" key={index}>

<a href={`/?categoryid=${category.categoryid}`}>

<div className="card card-background align-items-start mb-4 mb-lg-0">

<div

className="full-background"

style={{

backgroundImage: `url(${category.categoryimage})`,

backgroundSize: 'cover',

}}

></div>

<div className={`card-body ${classBody}`}>

<div className="d-block mt-10">

<p className="text-white font-weight-bold mb-1">{category.categoryname}</p>

<h4 className="text-white font-weight-bolder">{category.categoryname}</h4>

<a href={`/?categoryid=${category.categoryid}`} className="text-white text-sm font-weight-semibold mb-0">

Flere produkter &#62;

</a>

</div>

</div>

</div>

</a>

</div>

))}

</div>

);

}

import React from "react";

interface Props {

// order: string;

}

export default function KontaktPlade({

}: Props) {

return (

<>

<div className="container">

<div className="row mt-5">

<div className="col-12 col-md-4">

<div className="card shadow-xs border">

<div className="card-body p-3 d-flex">

<div className="icon icon-shape bg-dark text-white text-center border-radius-sm d-flex align-items-center justify-content-center">

<svg aria-hidden="true" xmlns="http://www.w3.org/2000/svg" width="24" height="24" fill="none" viewBox="0 0 24 24">

<path stroke="currentColor" stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="m4 12 8-8 8 8M6 10.5V19a1 1 0 0 0 1 1h3v-3a1 1 0 0 1 1-1h2a1 1 0 0 1 1 1v3h3a1 1 0 0 0 1-1v-8.5"/>

</svg>

</div>

<div className="ms-3">

<h5 className="mb-0">Adresse</h5>

<p className="mb-0 opacity-8">{import.meta.env.STOREADDRESS}</p>

</div>

</div>

</div>

</div>

<div className="col-12 col-md-4 my-4 my-md-0">

<div className="card shadow-xs border">

<div className="card-body p-3 d-flex">

<div className="icon icon-shape bg-dark text-white text-center border-radius-sm d-flex align-items-center justify-content-center">

<svg aria-hidden="true" xmlns="http://www.w3.org/2000/svg" width="24" height="24" fill="none" viewBox="0 0 24 24">

<path stroke="currentColor" stroke-linecap="round" stroke-width="2" d="M8 7V6a1 1 0 0 1 1-1h11a1 1 0 0 1 1 1v7a1 1 0 0 1-1 1h-1M3 18v-7a1 1 0 0 1 1-1h11a1 1 0 0 1 1 1v7a1 1 0 0 1-1 1H4a1 1 0 0 1-1-1Zm8-3.5a1.5 1.5 0 1 1-3 0 1.5 1.5 0 0 1 3 0Z"/>

</svg>

</div>

<div className="ms-3">

<h5 className="mb-0">Email</h5>

<p className="mb-0 opacity-8">{import.meta.env.STOREEMAIL}</p>

</div>

</div>

</div>

</div>

<div className="col-12 col-md-4">

<div className="card shadow-xs border">

<div className="card-body p-3 d-flex">

<div className="icon icon-shape bg-dark text-white text-center border-radius-sm d-flex align-items-center justify-content-center">

<svg aria-hidden="true" xmlns="http://www.w3.org/2000/svg" width="24" height="24" fill="none" viewBox="0 0 24 24">

<path stroke="currentColor" stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M18.427 14.768 17.2 13.542a1.733 1.733 0 0 0-2.45 0l-.613.613a1.732 1.732 0 0 1-2.45 0l-1.838-1.84a1.735 1.735 0 0 1 0-2.452l.612-.613a1.735 1.735 0 0 0 0-2.452L9.237 5.572a1.6 1.6 0 0 0-2.45 0c-3.223 3.2-1.702 6.896 1.519 10.117 3.22 3.221 6.914 4.745 10.12 1.535a1.601 1.601 0 0 0 0-2.456Z"/>

</svg>

</div>

<div className="ms-3">

<h5 className="mb-0">Telefon Nummer</h5>

<p className="mb-0 opacity-8">{import.meta.env.STOREPHONE}</p>

</div>

</div>

</div>

</div>

</div>

</div>

</>

);

};

import React, { useState, useEffect } from "react";

import { SendEmailToOwner } from "../../../FrontendRequests/Requests-Api/Email";

interface Props {}

export default function KontaktFormular({}: Props) {

// State for the input fields

const [email, setEmail] = useState("");

const [message, setMessage] = useState("");

const [subject, setSubject] = useState(""); // Added state for subject

const [isButtonDisabled, setIsButtonDisabled] = useState(true);

// Effect to check the fields and enable/disable the button

useEffect(() => {

checkFields();

}, [email, message, subject]); // Added subject as dependency

// Check fields to validate input

const checkFields = () => {

const isValid = message.trim() !== "" && email.trim() !== "" && subject.trim() !== "" && isValidEmail(email.trim());

setIsButtonDisabled(!isValid);

};

// Email validation function

const isValidEmail = (email: string) => {

const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

return emailRegex.test(email);

};

// Handle input change for email

const handleEmailChange = (event: React.ChangeEvent<HTMLInputElement>) => {

setEmail(event.target.value);

};

// Handle input change for message

const handleMessageChange = (event: React.ChangeEvent<HTMLTextAreaElement>) => {

setMessage(event.target.value);

};

// Handle input change for subject

const handleSubjectChange = (event: React.ChangeEvent<HTMLInputElement>) => {

setSubject(event.target.value);

};

// Handle form submission (you can add your own logic here)

const handleSubmit = (event: React.FormEvent) => {

event.preventDefault();

if (message.trim() !== "" && email.trim() !== "" && subject.trim() !== "") {

SendEmailToOwner(email, message, subject).then((result) => {

if(result.result) {

alert("Email leveret! Tak for du kontaktede os")

window.location.href = "Kontakt"

} else {

}

})

// You can replace the console.log with an actual submission logic.

} else {

console.log("Invalid form input.");

}

};

return (

<section className="bg-gray-100 d-flex justify-content-center align-items-center">

<div className="row w-100">

<div className="col-12 col-lg-6 p-3 p-md-5 mx-auto">

<h5 className="mb-4">Kontakt information</h5>

<form onSubmit={handleSubmit}>

<div className="form-group">

<label>Email address</label>

<input

id="emailbox"

type="email"

className="form-control"

placeholder="Enter your email address"

value={email}

onChange={handleEmailChange}

/>

</div>

<div className="form-group">

<label>Emne</label> {/\* Added subject label \*/}

<input

id="subjectbox"

type="text"

className="form-control"

placeholder="Enter the subject"

value={subject}

onChange={handleSubjectChange} // Added subject handler

/>

</div>

<div className="form-group">

<label>Besked</label>

<textarea

id="messagebox"

className="form-control"

placeholder="Skriv besked her"

style={{ width: "100%", height: "150px" }}

value={message}

onChange={handleMessageChange}

></textarea>

</div>

<button

className="btn btn-dark w-100 mt-4"

id="kontaktbutton"

type="submit"

disabled={isButtonDisabled}

>

Fortsæt

</button>

</form>

</div>

</div>

</section>

);

}

import React, {useEffect, useState } from 'react';

import ProductCartItem from './productCartItem';

import { useNotifications } from '../NotifcationSystem/NotificationContext';

import { GetProduct } from '../../../FrontendRequests/Requests-Api/Product';

import type {ExtendedBasketProduct} from "../NotifcationSystem/NotificationContext";

export default function ShoppingCart({

}) {

const { Basket, removeFromBasket } = useNotifications();

const [GetProducts, SetProducts] = useState<ExtendedBasketProduct[] | null>(null);

const [checkoutdisabled, setcheckoutdisabled] = useState<boolean>(true);

const GetProductsByIds = async () => {

try {

const productPromises = Basket.map(async (product, index) => {

const response = await GetProduct(product.ProductId.toString()); // Fetch product by ID

if (response.result) { // Check if response.result is truthy

return {

...response.data, // Spread the rest of the product data

productcolorsizes: [product.ProductCustoms],

quantity: product.quantity,

};

} else {

removeFromBasket(index); // Remove the product from the basket

return null; // Return null for invalid products

}

});

// Wait for all promises to resolve

const updatedProducts = await Promise.all(productPromises);

// Filter out any null values and update the products

const validProducts = updatedProducts.filter(product => product !== null);

SetProducts(validProducts); // Assuming SetProducts updates the state

} catch (error) {

console.error('Error fetching products:', error);

}

};

useEffect(() => {

GetProductsByIds();

setcheckoutdisabled(Basket.length === 0);

}, []);

return (

<>

<div className="container mt-5">

<h2 className="mb-3 text-center">Indkøbskurv</h2>

<h5 className="text-center mb-5">Du har {GetProducts?.length} produkter i din indkøbskurv</h5>

<div className="row">

<div className="col-12 col-lg-7">

{GetProducts?.map((product, i) =>

<>

{<hr className="horizontal dark my-4" />

}

<ProductCartItem

index={i}

ProductId={product.productid}

image\_url={product.images[0].image\_url} // First image URL

image\_desc={product.images[0].created\_at || ''} // Image description (optional)

ProductName={product.name} // Product name

ProductColor={product.productcolorsizes[0].colorname || ''} // First color name

ProductSize={product.productcolorsizes[0].sizename || ''} // First size name

ProductPrice={(Number(product.salgpris\_ex\_moms) + Number(product.udgående\_moms))} // Product price

ProductStock={product.productcolorsizes[0].quantity} // Product stock (quantity of the first color/size combination)

ProductQuantity={product.quantity}

ProductColorSizes={product.productcolorsizes[0]}

/>

</>

)}

</div>

<div className="col-12 col-lg-5 mt-5 mt-lg-0">

<div className="card shadow-xs border bg-gray-100">

<div className="card-body p-lg-5">

<h5 className="mb-4">Indkøbskurv</h5>

<button

name="Checkout"

className="form-control btn btn-dark btn-lg w-100"

style={{ color: "yellow" }}

disabled={checkoutdisabled}

onClick={() => window.location.href = `/Checkout`}

>

Kasse

</button>

<a href='/Butik' className="btn btn-white btn-lg w-100">Shop videre</a>

<p className="text-center">Moms inkludered i kvitteringen.</p>

</div>

</div>

<p className="d-flex align-items-center justify-content-center text-body mt-4">

<svg className="me-2" width="11" height="13" viewBox="0 0 11 13" fill="none" xmlns="http://www.w3.org/2000/svg">

<path d="M3.21652 6.14286H7.78795V4.42857C7.78795 3.79762 7.56473 3.25893 7.1183 2.8125C6.67188 2.36607 6.13318 2.14286 5.50223 2.14286C4.87128 2.14286 4.33259 2.36607 3.88616 2.8125C3.43973 3.25893 3.21652 3.79762 3.21652 4.42857V6.14286ZM10.6451 7V12.1429C10.6451 12.381 10.5618 12.5833 10.3951 12.75C10.2284 12.9167 10.026 13 9.78795 13H1.21652C0.978423 13 0.776042 12.9167 0.609375 12.75C0.442708 12.5833 0.359375 12.381 0.359375 12.1429V7C0.359375 6.7619 0.442708 6.55952 0.609375 6.39286C0.776042 6.22619 0.978423 6.14286 1.21652 6.14286H1.50223V4.42857C1.50223 3.33333 1.89509 2.39286 2.6808 1.60714C3.46652 0.821428 4.40699 0.428571 5.50223 0.428571C6.59747 0.428571 7.53795 0.821428 8.32366 1.60714C9.10938 2.39286 9.50223 3.33333 9.50223 4.42857V6.14286H9.78795C10.026 6.14286 10.2284 6.22619 10.3951 6.39286C10.5618 6.55952 10.6451 6.7619 10.6451 7Z" fill="#495057"/>

</svg>

Secured Payment with: Stripe

</p>

<div className="d-flex justify-content-center">

<svg className="me-2" width="31" height="21" viewBox="0 0 31 21" fill="none" xmlns="http://www.w3.org/2000/svg">

<path d="M26.5173 11.6875H24.6691C24.7941 11.3571 25.0887 10.558 25.553 9.29018L25.5932 9.16964C25.6289 9.08036 25.6735 8.96429 25.7271 8.82143C25.7807 8.67857 25.8209 8.5625 25.8477 8.47321L26.0084 9.20982L26.5173 11.6875ZM7.17801 10.817L6.40123 6.86607C6.30301 6.38393 5.96819 6.14286 5.39676 6.14286H1.80748L1.78069 6.31696C4.55748 7.02232 6.35659 8.52232 7.17801 10.817ZM9.57534 6.14286L7.40569 12.0089L7.17801 10.817C6.94587 10.192 6.56641 9.61607 6.03962 9.08929C5.51284 8.55357 4.92801 8.15625 4.28516 7.89732L6.09319 14.7277H8.43694L11.9325 6.14286H9.57534ZM11.4369 14.7411H13.6602L15.053 6.14286H12.8298L11.4369 14.7411ZM21.7227 6.35714C21.1066 6.11607 20.4414 5.99553 19.7271 5.99553C18.6289 5.99553 17.7316 6.25893 17.0352 6.78571C16.3387 7.3125 15.986 7.99554 15.9771 8.83482C15.9682 9.74554 16.6155 10.5223 17.9191 11.1652C18.3477 11.3705 18.6468 11.5536 18.8164 11.7143C18.9861 11.875 19.0709 12.0491 19.0709 12.2366C19.0709 12.5045 18.9369 12.7098 18.6691 12.8527C18.4012 12.9955 18.0932 13.067 17.745 13.067C16.9771 13.067 16.2807 12.9196 15.6557 12.625L15.3611 12.4777L15.053 14.4062C15.7137 14.7098 16.5396 14.8616 17.5307 14.8616C18.6914 14.8705 19.62 14.6071 20.3164 14.0714C21.0218 13.5357 21.3834 12.8214 21.4012 11.9286C21.4012 10.9821 20.7762 10.2054 19.5262 9.59821C19.0887 9.375 18.7718 9.1875 18.5753 9.03571C18.3789 8.88393 18.2807 8.71429 18.2807 8.52678C18.2807 8.33036 18.3878 8.16071 18.6021 8.01786C18.8253 7.86607 19.1423 7.79018 19.553 7.79018C20.178 7.78125 20.7316 7.88839 21.2137 8.11161L21.4146 8.21875L21.7227 6.35714ZM27.4146 6.14286H25.7003C25.12 6.14286 24.7316 6.38393 24.5352 6.86607L21.2405 14.7411H23.5709L24.0396 13.4554H26.8789C26.9235 13.6518 27.0128 14.0804 27.1468 14.7411H29.2093L27.4146 6.14286ZM30.9236 1.85714V19C30.9236 19.4643 30.7539 19.8661 30.4146 20.2054C30.0753 20.5446 29.6736 20.7143 29.2093 20.7143H1.78069C1.31641 20.7143 0.914621 20.5446 0.575335 20.2054C0.236049 19.8661 0.0664062 19.4643 0.0664062 19V1.85714C0.0664062 1.39286 0.236049 0.991071 0.575335 0.651785C0.914621 0.312499 1.31641 0.142856 1.78069 0.142856H29.2093C29.6736 0.142856 30.0753 0.312499 30.4146 0.651785C30.7539 0.991071 30.9236 1.39286 30.9236 1.85714Z" fill="#495057"/>

</svg>

<svg className="me-2" width="31" height="21" viewBox="0 0 31 21" fill="none" xmlns="http://www.w3.org/2000/svg">

<path d="M15.053 2.99553C13.9102 2.23661 12.6557 1.85714 11.2896 1.85714C10.37 1.85714 9.48605 2.04018 8.63784 2.40625C7.79855 2.76339 7.07534 3.24553 6.46819 3.85268C5.86105 4.45982 5.37444 5.18304 5.00837 6.02232C4.65123 6.86161 4.47266 7.74107 4.47266 8.66071C4.47266 9.58929 4.65123 10.4732 5.00837 11.3125C5.37444 12.1518 5.86105 12.875 6.46819 13.4821C7.07534 14.0893 7.79855 14.5759 8.63784 14.942C9.47712 15.2991 10.361 15.4777 11.2896 15.4777C12.6557 15.4777 13.9102 15.0982 15.053 14.3393C13.8834 13.3839 13.0887 12.2009 12.6691 10.7902C12.2494 9.37054 12.2494 7.95536 12.6691 6.54464C13.0977 5.13393 13.8923 3.95089 15.053 2.99553ZM15.495 3.31696C14.37 4.20089 13.6021 5.31696 13.1914 6.66518C12.7807 8.00446 12.7762 9.34375 13.178 10.683C13.5887 12.0223 14.361 13.1339 15.495 14.0179C16.6289 13.1339 17.3968 12.0223 17.7986 10.683C18.2093 9.34375 18.2093 8.00446 17.7986 6.66518C17.3878 5.31696 16.62 4.20089 15.495 3.31696ZM15.9369 2.99553C17.0977 3.95089 17.8878 5.13839 18.3075 6.55803C18.736 7.96875 18.7405 9.38393 18.3209 10.8036C17.9012 12.2143 17.1066 13.3929 15.9369 14.3393C17.0798 15.0982 18.3343 15.4777 19.7003 15.4777C20.6289 15.4777 21.5128 15.2991 22.3521 14.942C23.1914 14.5759 23.9146 14.0893 24.5218 13.4821C25.1289 12.875 25.6111 12.1518 25.9682 11.3125C26.3343 10.4732 26.5173 9.58929 26.5173 8.66071C26.5173 7.74107 26.3343 6.86161 25.9682 6.02232C25.6111 5.18304 25.1289 4.45982 24.5218 3.85268C23.9146 3.24553 23.1869 2.76339 22.3387 2.40625C21.4994 2.04018 20.62 1.85714 19.7003 1.85714C18.3343 1.85714 17.0798 2.23661 15.9369 2.99553ZM25.8611 12.6652H25.9548V12.625H25.7271V12.6652H25.8209V12.8929H25.8611V12.6652ZM26.2494 12.8929H26.303V12.625H26.2361L26.1557 12.7991L26.0753 12.625H26.0084V12.8929H26.0486V12.692L26.1289 12.8661H26.1825L26.2494 12.692V12.8929ZM26.1423 18.7857V18.8125H26.1155H26.0753V18.7723H26.1155H26.1423V18.7857ZM26.1423 18.9062H26.1825L26.1289 18.8393C26.1378 18.8393 26.1468 18.8393 26.1557 18.8393L26.1691 18.8259C26.178 18.817 26.1825 18.8036 26.1825 18.7857C26.1825 18.7679 26.178 18.7545 26.1691 18.7455L26.1557 18.7321C26.1378 18.7321 26.1244 18.7321 26.1155 18.7321H26.0352V18.9062H26.0753V18.8393H26.0887L26.1423 18.9062ZM9.38784 18.4107C9.28962 18.3036 9.24051 18.1652 9.24051 17.9955C9.24051 17.8259 9.28962 17.6875 9.38784 17.5804C9.48605 17.4732 9.61998 17.4196 9.78962 17.4196C9.95034 17.4196 10.0798 17.4777 10.178 17.5937C10.2762 17.7009 10.3253 17.8348 10.3253 17.9955C10.3253 18.1652 10.2762 18.3036 10.178 18.4107C10.0798 18.5179 9.95034 18.5714 9.78962 18.5714C9.61998 18.5714 9.48605 18.5179 9.38784 18.4107ZM15.1066 17.8348C15.1512 17.5491 15.3075 17.4062 15.5753 17.4062C15.8432 17.4062 15.9994 17.5491 16.0441 17.8348H15.1066ZM20.4905 18.4107C20.3923 18.3036 20.3432 18.1652 20.3432 17.9955C20.3432 17.8259 20.3923 17.6875 20.4905 17.5804C20.5887 17.4732 20.7182 17.4196 20.8789 17.4196C21.0396 17.4196 21.1691 17.4777 21.2673 17.5937C21.3744 17.7009 21.428 17.8348 21.428 17.9955C21.428 18.1652 21.3789 18.3036 21.2807 18.4107C21.1825 18.5179 21.0486 18.5714 20.8789 18.5714C20.7182 18.5714 20.5887 18.5179 20.4905 18.4107ZM24.1334 18.4107C24.0352 18.2946 23.9861 18.1562 23.9861 17.9955C23.9861 17.8348 24.0352 17.7009 24.1334 17.5937C24.2405 17.4777 24.3744 17.4196 24.5352 17.4196C24.6959 17.4196 24.8253 17.4777 24.9236 17.5937C25.0307 17.7009 25.0843 17.8348 25.0843 17.9955C25.0843 18.1652 25.0307 18.3036 24.9236 18.4107C24.8253 18.5179 24.6959 18.5714 24.5352 18.5714C24.3744 18.5714 24.2405 18.5179 24.1334 18.4107ZM26.1021 18.9598C26.0843 18.9598 26.0664 18.9554 26.0486 18.9464C26.0396 18.9464 26.0262 18.9375 26.0084 18.9196C25.9905 18.9018 25.9816 18.8884 25.9816 18.8795C25.9727 18.8616 25.9682 18.8437 25.9682 18.8259C25.9682 18.7991 25.9727 18.7812 25.9816 18.7723C25.9816 18.7545 25.9905 18.7366 26.0084 18.7187C26.0173 18.7098 26.0218 18.7054 26.0218 18.7054C26.0396 18.7054 26.0486 18.7009 26.0486 18.692C26.0664 18.683 26.0843 18.6786 26.1021 18.6786C26.1289 18.6786 26.1468 18.683 26.1557 18.692C26.1914 18.7098 26.2093 18.7187 26.2093 18.7187L26.2361 18.7723V18.7857C26.245 18.8036 26.2494 18.817 26.2494 18.8259L26.2361 18.8393C26.2361 18.8571 26.2361 18.8705 26.2361 18.8795C26.2361 18.8884 26.2316 18.8929 26.2227 18.8929L26.2093 18.9196C26.1914 18.9375 26.1736 18.9464 26.1557 18.9464C26.1468 18.9554 26.1289 18.9598 26.1021 18.9598ZM8.08873 18.9062H8.49051V17.7679C8.49051 17.5536 8.42355 17.3839 8.28962 17.2589C8.16462 17.125 7.99051 17.0536 7.7673 17.0446C7.48159 17.0446 7.27176 17.1518 7.13784 17.3661C7.01284 17.1518 6.81194 17.0446 6.53516 17.0446C6.32087 17.0446 6.14676 17.1339 6.01284 17.3125V17.0982H5.61105V18.9062H6.01284V17.9018C6.01284 17.5804 6.16016 17.4196 6.4548 17.4196C6.72266 17.4196 6.85659 17.5804 6.85659 17.9018V18.9062H7.24498V17.9018C7.24498 17.5804 7.3923 17.4196 7.68694 17.4196C7.9548 17.4196 8.08873 17.5804 8.08873 17.9018V18.9062ZM10.3119 18.9062H10.7003V17.9955V17.0982H10.3119V17.3125C10.1602 17.1339 9.96819 17.0446 9.73605 17.0446C9.47712 17.0446 9.26284 17.1339 9.09319 17.3125C8.92355 17.4911 8.83873 17.7187 8.83873 17.9955C8.83873 18.2723 8.92355 18.5 9.09319 18.6786C9.26284 18.8571 9.47712 18.9464 9.73605 18.9464C9.98605 18.9464 10.178 18.8571 10.3119 18.6786V18.9062ZM12.6959 18.3571C12.6959 18.0536 12.486 17.875 12.0664 17.8214L11.8789 17.7946C11.6735 17.7589 11.5709 17.6964 11.5709 17.6071C11.5709 17.4732 11.6825 17.4062 11.9057 17.4062C12.1111 17.4062 12.303 17.4554 12.4816 17.5536L12.6423 17.2321C12.4459 17.1071 12.2003 17.0446 11.9057 17.0446C11.6735 17.0446 11.4905 17.0982 11.3566 17.2054C11.2227 17.3125 11.1557 17.4554 11.1557 17.6339C11.1557 17.9286 11.3655 18.1027 11.7852 18.1562L11.9593 18.183C12.1735 18.2187 12.2807 18.2812 12.2807 18.3705C12.2807 18.5223 12.1423 18.5982 11.8655 18.5982C11.6423 18.5982 11.4414 18.5357 11.2628 18.4107L11.0887 18.7187C11.3119 18.8705 11.5709 18.9464 11.8655 18.9464C12.1244 18.9464 12.3253 18.8929 12.4682 18.7857C12.62 18.6786 12.6959 18.5357 12.6959 18.3571ZM14.4369 18.8125L14.3298 18.4777C14.2137 18.5402 14.0977 18.5714 13.9816 18.5714C13.8119 18.5714 13.7271 18.4732 13.7271 18.2768V17.4598H14.37V17.0982H13.7271V16.5491H13.3253V17.0982H12.9503V17.4598H13.3253V18.2768C13.3253 18.7232 13.5352 18.9464 13.9548 18.9464C14.1423 18.9464 14.303 18.9018 14.4369 18.8125ZM15.5887 17.0446C15.3298 17.0446 15.1155 17.1339 14.9459 17.3125C14.7762 17.4911 14.6914 17.7187 14.6914 17.9955C14.6914 18.2812 14.7762 18.5134 14.9459 18.692C15.1244 18.8616 15.3477 18.9464 15.6155 18.9464C15.9102 18.9464 16.1557 18.8616 16.3521 18.692L16.1646 18.3973C16.0039 18.5312 15.8298 18.5982 15.6423 18.5982C15.3387 18.5982 15.1557 18.4509 15.0932 18.1562H16.4459V17.9955C16.4459 17.7098 16.3655 17.4821 16.2048 17.3125C16.0441 17.1339 15.8387 17.0446 15.5887 17.0446ZM17.7182 17.0446C17.5128 17.0446 17.3566 17.1339 17.2494 17.3125V17.0982H16.8477V18.9062H17.2494V17.8884C17.2494 17.5759 17.3789 17.4196 17.6378 17.4196C17.7271 17.4196 17.8075 17.4375 17.8789 17.4732L17.9994 17.0982C17.9191 17.0625 17.8253 17.0446 17.7182 17.0446ZM18.3744 17.3125C18.2048 17.4911 18.12 17.7187 18.12 17.9955C18.12 18.2723 18.2048 18.5 18.3744 18.6786C18.553 18.8571 18.7896 18.9464 19.0843 18.9464C19.3432 18.9464 19.5575 18.875 19.7271 18.7321L19.5396 18.4107C19.3789 18.5268 19.2227 18.5804 19.0709 18.5714C18.9102 18.5714 18.7762 18.5179 18.6691 18.4107C18.5709 18.3036 18.5218 18.1652 18.5218 17.9955C18.5218 17.8259 18.5709 17.6875 18.6691 17.5804C18.7762 17.4732 18.9102 17.4196 19.0709 17.4196C19.2405 17.4196 19.3968 17.4732 19.5396 17.5804L19.7271 17.2589C19.5485 17.1161 19.3343 17.0446 19.0843 17.0446C18.7896 17.0446 18.553 17.1339 18.3744 17.3125ZM21.4012 18.9062H21.803V17.9955V17.0982H21.4012V17.3125C21.2673 17.1339 21.0798 17.0446 20.8387 17.0446C20.5798 17.0446 20.361 17.1339 20.1825 17.3125C20.0128 17.4911 19.928 17.7187 19.928 17.9955C19.928 18.2723 20.0128 18.5 20.1825 18.6786C20.361 18.8571 20.5798 18.9464 20.8387 18.9464C21.0887 18.9464 21.2762 18.8571 21.4012 18.6786V18.9062ZM23.1825 17.0446C22.9771 17.0446 22.8209 17.1339 22.7137 17.3125V17.0982H22.3253V18.9062H22.7137V17.8884C22.7137 17.5759 22.8432 17.4196 23.1021 17.4196C23.1914 17.4196 23.2718 17.4375 23.3432 17.4732L23.4637 17.0982C23.3923 17.0625 23.2986 17.0446 23.1825 17.0446ZM25.0575 18.9062H25.4459V17.9955V16.3616H25.0575V17.3125C24.9236 17.1339 24.7316 17.0446 24.4816 17.0446C24.2316 17.0446 24.0173 17.1384 23.8387 17.3259C23.6691 17.5045 23.5843 17.7277 23.5843 17.9955C23.5843 18.2634 23.6691 18.4911 23.8387 18.6786C24.0173 18.8571 24.2316 18.9464 24.4816 18.9464C24.7405 18.9464 24.9325 18.8571 25.0575 18.6786V18.9062ZM26.1021 18.6384L26.0753 18.6518C26.0575 18.6518 26.0441 18.6518 26.0352 18.6518C26.0173 18.6607 25.9994 18.6741 25.9816 18.692C25.9548 18.7009 25.9414 18.7187 25.9414 18.7455C25.9325 18.7634 25.928 18.7902 25.928 18.8259C25.928 18.8527 25.9325 18.875 25.9414 18.8929C25.9414 18.9107 25.9548 18.9286 25.9816 18.9464C25.9994 18.9643 26.0173 18.9777 26.0352 18.9866C26.053 18.9955 26.0753 19 26.1021 19C26.1378 19 26.1646 18.9955 26.1825 18.9866C26.1825 18.9777 26.1914 18.9687 26.2093 18.9598L26.2361 18.9464C26.245 18.9375 26.2584 18.9196 26.2762 18.8929C26.2852 18.875 26.2896 18.8527 26.2896 18.8259C26.2896 18.7902 26.2852 18.7634 26.2762 18.7455C26.2673 18.7366 26.2539 18.7187 26.2361 18.692C26.2361 18.683 26.2271 18.6741 26.2093 18.6652L26.1825 18.6518C26.1736 18.6518 26.1602 18.6518 26.1423 18.6518C26.1244 18.6429 26.111 18.6384 26.1021 18.6384ZM30.9236 1.85714V19C30.9236 19.4643 30.7539 19.8661 30.4146 20.2054C30.0753 20.5446 29.6736 20.7143 29.2093 20.7143H1.78069C1.31641 20.7143 0.914621 20.5446 0.575335 20.2054C0.236049 19.8661 0.0664062 19.4643 0.0664062 19V1.85714C0.0664062 1.39286 0.236049 0.991071 0.575335 0.651785C0.914621 0.312499 1.31641 0.142856 1.78069 0.142856H29.2093C29.6736 0.142856 30.0753 0.312499 30.4146 0.651785C30.7539 0.991071 30.9236 1.39286 30.9236 1.85714Z" fill="#495057"/>

</svg>

<svg className="me-2" width="31" height="21" viewBox="0 0 31 21" fill="none" xmlns="http://www.w3.org/2000/svg">

<path d="M1.66016 7.5625H2.85212L2.24944 6.11607L1.66016 7.5625ZM9.97712 14.6071L10.9682 13.5491L10.0307 12.4911H7.84766V13.1473H9.74944V13.8839H7.84766V14.6071H9.97712ZM12.0932 13.5625L13.4191 15.0357V12.1295L12.0932 13.5625ZM15.9503 12.933C15.9503 12.6384 15.7718 12.4911 15.4146 12.4911H14.2896V13.4152H15.4012C15.7673 13.4152 15.9503 13.2545 15.9503 12.933ZM19.8209 12.8795C19.8209 12.6205 19.6334 12.4911 19.2584 12.4911H18.1602V13.308H19.245C19.6289 13.308 19.8209 13.1652 19.8209 12.8795ZM16.0977 6.63839C16.0977 6.37946 15.9102 6.25 15.5352 6.25H14.4369V7.05357H15.5218C15.9057 7.05357 16.0977 6.91518 16.0977 6.63839ZM22.245 7.5625H23.4369L22.8477 6.11607L22.245 7.5625ZM9.42801 5.48661V9.11607H8.54409V6.27678L7.28516 9.11607H6.52176L5.26284 6.27678V9.11607H3.49498L3.16016 8.3125H1.35212L1.0173 9.11607H0.0797991L1.63337 5.48661H2.91909L4.3923 8.92857V5.48661H5.81194L6.95034 7.95089L7.98159 5.48661H9.42801ZM16.8744 12.933C16.8744 13.1116 16.8477 13.2679 16.7941 13.4018C16.7494 13.5357 16.6869 13.6473 16.6066 13.7366C16.5352 13.8259 16.4369 13.9018 16.3119 13.9643C16.1869 14.0179 16.0709 14.0625 15.9637 14.0982C15.8566 14.125 15.7137 14.1429 15.5352 14.1518C15.3655 14.1607 15.2271 14.1652 15.12 14.1652C15.0128 14.1652 14.8655 14.1652 14.678 14.1652C14.4994 14.1562 14.37 14.1518 14.2896 14.1518V15.3705H12.6021L11.5307 14.1652L10.4191 15.3705H6.99051V11.7411H10.4727L11.5441 12.933L12.6423 11.7411H15.4146C16.3878 11.7411 16.8744 12.1384 16.8744 12.933ZM12.9771 8.36607V9.11607H10.0709V5.48661H12.9771V6.25H10.9414V6.90625H12.9236V7.64286H10.9414V8.36607H12.9771ZM30.9236 15.8527V18.9196C30.9236 19.4107 30.7494 19.8304 30.4012 20.1786C30.0619 20.5357 29.6468 20.7143 29.1557 20.7143H1.83426C1.34319 20.7143 0.923549 20.5357 0.575335 20.1786C0.236049 19.8304 0.0664062 19.4107 0.0664062 18.9196V9.83929H1.55301L1.88783 9.02232H2.62444L2.95926 9.83929H5.87891V9.22321L6.13337 9.83929H7.64676L7.91462 9.20982V9.83929H15.1602V8.51339L15.2941 8.5C15.3834 8.5 15.428 8.5625 15.428 8.6875V9.83929H19.1646V9.53125C19.37 9.63839 19.6155 9.71875 19.9012 9.77232C20.1869 9.82589 20.4191 9.85714 20.5977 9.86607C20.7852 9.86607 21.0664 9.86161 21.4414 9.85268C21.8253 9.84375 22.0575 9.83929 22.1378 9.83929L22.4727 9.02232H23.2227L23.5575 9.83929H26.5977V9.0625L27.053 9.83929H29.4905V4.77678H27.0798V5.36607L26.745 4.77678H24.2673V5.36607L23.9593 4.77678H20.6244C20.0084 4.77678 19.5218 4.875 19.1646 5.07143V4.77678H16.8611V5.07143C16.6468 4.875 16.3209 4.77678 15.8834 4.77678H7.47266L6.89676 6.07589L6.32087 4.77678H3.66909V5.36607L3.37444 4.77678H1.11105L0.0664062 7.17411V1.9375C0.0664062 1.44643 0.236049 1.02679 0.575335 0.678571C0.923549 0.321428 1.34319 0.142856 1.83426 0.142856H29.1557C29.6468 0.142856 30.0619 0.321428 30.4012 0.678571C30.7494 1.02679 30.9236 1.44643 30.9236 1.9375V11.0179H29.3164C28.861 11.0179 28.4994 11.1161 28.2316 11.3125V11.0179H25.8611C25.37 11.0179 25.0218 11.1161 24.8164 11.3125V11.0179H20.5843V11.3125C20.3075 11.1161 19.9191 11.0179 19.4191 11.0179H16.62V11.3125C16.4146 11.1161 16.0084 11.0179 15.4012 11.0179H12.2673L11.5441 11.7946L10.8744 11.0179H6.20034V16.0804H10.7941L11.5307 15.2902L12.2271 16.0804H15.053V14.8884H15.3343C15.861 14.8884 16.2628 14.8304 16.5396 14.7143V16.0804H18.87V14.7545H18.9771C19.0485 14.7545 19.0932 14.7634 19.1111 14.7812C19.1289 14.7991 19.1378 14.8437 19.1378 14.9152V16.0804H26.2227C26.7316 16.0804 27.1244 15.9732 27.4012 15.7589V16.0804H29.6512C30.1869 16.0804 30.6111 16.0045 30.9236 15.8527ZM20.7718 12.7188C20.7718 12.9241 20.7182 13.1161 20.6111 13.2946C20.5039 13.4732 20.3521 13.6027 20.1557 13.683C20.3789 13.7634 20.5307 13.8795 20.6111 14.0312C20.6914 14.183 20.7316 14.3884 20.7316 14.6473V15.3705H19.8611V14.7679C19.8611 14.4732 19.8075 14.2812 19.7003 14.192C19.5932 14.0937 19.3878 14.0446 19.0843 14.0446H18.1602V15.3705H17.2896V11.7411H19.3521C19.7807 11.7411 20.1244 11.808 20.3834 11.942C20.6423 12.0759 20.7718 12.3348 20.7718 12.7188ZM17.0619 6.46428C17.0619 6.67857 17.0039 6.875 16.8878 7.05357C16.7807 7.23214 16.6334 7.36161 16.4459 7.44196C16.678 7.52232 16.8298 7.63839 16.9012 7.79018C16.9816 7.93304 17.0218 8.13839 17.0218 8.40625V9.11607H16.1512C16.1512 9.03571 16.1512 8.91964 16.1512 8.76786C16.1602 8.60714 16.1602 8.49553 16.1512 8.43303C16.1512 8.36161 16.1378 8.27679 16.1111 8.17857C16.0932 8.08036 16.0575 8.00893 16.0039 7.96428C15.9503 7.91964 15.87 7.88393 15.7628 7.85714C15.6646 7.82143 15.5352 7.80357 15.3744 7.80357H14.4369V9.11607H13.5798V5.48661L15.6289 5.5C16.0664 5.5 16.4146 5.56696 16.6736 5.70089C16.9325 5.82589 17.0619 6.08036 17.0619 6.46428ZM24.1468 14.6205V15.3705H21.2539V11.7411H24.1468V12.4911H22.1244V13.1473H24.1066V13.8839H22.1244V14.6071L24.1468 14.6205ZM18.4414 5.48661V9.11607H17.5575V5.48661H18.4414ZM27.7227 14.2187C27.7227 14.9866 27.2673 15.3705 26.3566 15.3705H24.6691V14.5937H26.3566C26.6602 14.5937 26.8119 14.4821 26.8119 14.2589C26.8119 14.1161 26.7361 14.0223 26.5843 13.9777C26.4325 13.933 26.245 13.9107 26.0218 13.9107C25.8075 13.9107 25.5887 13.8973 25.3655 13.8705C25.1423 13.8348 24.9548 13.7321 24.803 13.5625C24.6512 13.3929 24.5753 13.1473 24.5753 12.8259C24.5753 12.4777 24.6914 12.2098 24.9236 12.0223C25.1557 11.8348 25.4503 11.7411 25.8075 11.7411H27.5486V12.5045H25.9548C25.6334 12.5045 25.4727 12.6161 25.4727 12.8393C25.4727 12.9821 25.5485 13.0759 25.7003 13.1205C25.861 13.1562 26.0485 13.1741 26.2628 13.1741C26.486 13.1652 26.7048 13.1741 26.9191 13.2009C27.1423 13.2277 27.3298 13.3259 27.4816 13.4955C27.6423 13.6562 27.7227 13.8973 27.7227 14.2187ZM30.9236 13.5491V14.9018C30.7093 15.2143 30.3164 15.3705 29.745 15.3705H28.0709V14.5937H29.745C30.0396 14.5937 30.1869 14.4821 30.1869 14.2589C30.1869 14.1429 30.1289 14.058 30.0128 14.0045C29.9057 13.9509 29.7673 13.9286 29.5977 13.9375C29.4369 13.9375 29.2584 13.9286 29.0619 13.9107C28.8744 13.8839 28.6959 13.8482 28.5262 13.8036C28.3655 13.75 28.2271 13.6429 28.1111 13.4821C28.0039 13.3125 27.9503 13.0937 27.9503 12.8259C27.9503 12.4777 28.0664 12.2098 28.2986 12.0223C28.5396 11.8348 28.8387 11.7411 29.1959 11.7411H30.9236V12.5045H29.3432C29.0218 12.5045 28.8611 12.6161 28.8611 12.8393C28.8611 13.0179 28.9905 13.1161 29.2494 13.1339C29.5084 13.1518 29.8119 13.1741 30.1602 13.2009C30.5173 13.2277 30.7718 13.3437 30.9236 13.5491ZM28.7137 5.5V9.11607H27.4816L25.8477 6.39732V9.11607H24.0798L23.7316 8.3125H21.9369L21.6021 9.11607H20.5977C19.4459 9.11607 18.87 8.52232 18.87 7.33482C18.87 6.10268 19.4637 5.48661 20.6512 5.48661H21.495V6.27678C21.4325 6.27678 21.3075 6.27232 21.12 6.26339C20.9325 6.25446 20.803 6.25446 20.7316 6.26339C20.6691 6.26339 20.5664 6.27232 20.4236 6.29018C20.2896 6.29911 20.1959 6.32589 20.1423 6.37053C20.0887 6.41518 20.0218 6.47768 19.9414 6.55803C19.87 6.62946 19.8209 6.73214 19.7941 6.86607C19.7673 6.99107 19.7539 7.13839 19.7539 7.30803C19.7539 7.64732 19.8119 7.90625 19.928 8.08482C20.053 8.26339 20.2762 8.35268 20.5977 8.35268H20.9861L22.2182 5.5H23.5173L24.9771 8.92857V5.5H26.303L27.8298 8.01786V5.5H28.7137Z" fill="#495057"/>

</svg>

<svg className="me-2" width="31" height="21" viewBox="0 0 31 21" fill="none" xmlns="http://www.w3.org/2000/svg">

<path d="M10.0441 10.5625C10.0441 10.8929 9.92801 11.1696 9.69587 11.3929C9.47266 11.6071 9.19587 11.7143 8.86551 11.7143C8.60659 11.7143 8.39676 11.6429 8.23605 11.5C8.08426 11.3571 8.00837 11.1607 8.00837 10.9107C8.00837 10.5804 8.11998 10.3036 8.34319 10.0804C8.56641 9.84821 8.84319 9.73214 9.17355 9.73214C9.42355 9.73214 9.62891 9.80804 9.78962 9.95982C9.95926 10.1027 10.0441 10.3036 10.0441 10.5625ZM20.5575 8.56696C20.5575 8.94196 20.4593 9.19643 20.2628 9.33036C20.0664 9.46429 19.7718 9.53125 19.3789 9.53125L18.9503 9.54464L19.178 8.11161C19.1959 8.01339 19.2539 7.96428 19.3521 7.96428H19.5932C19.7896 7.96428 19.9459 7.97321 20.0619 7.99107C20.178 8.00893 20.2896 8.06696 20.3968 8.16518C20.5039 8.25446 20.5575 8.38839 20.5575 8.56696ZM25.2584 10.5625C25.2584 10.8839 25.1423 11.1562 24.9102 11.3795C24.6869 11.6027 24.4146 11.7143 24.0932 11.7143C23.8343 11.7143 23.6244 11.6429 23.4637 11.5C23.303 11.3571 23.2227 11.1607 23.2227 10.9107C23.2227 10.5804 23.3343 10.3036 23.5575 10.0804C23.7807 9.84821 24.0575 9.73214 24.3878 9.73214C24.6378 9.73214 24.8432 9.80804 25.0039 9.95982C25.1735 10.1027 25.2584 10.3036 25.2584 10.5625ZM6.93694 8.27232C6.93694 7.74554 6.76284 7.36607 6.41462 7.13393C6.07534 6.89286 5.62891 6.77232 5.07534 6.77232H2.93248C2.76283 6.77232 2.66908 6.85714 2.65123 7.02678L1.78069 12.4911C1.77176 12.5446 1.78516 12.5937 1.82087 12.6384C1.85658 12.683 1.90123 12.7054 1.9548 12.7054H2.97266C3.15123 12.7054 3.24944 12.6205 3.2673 12.4509L3.50837 10.9777C3.5173 10.9062 3.54855 10.8482 3.60212 10.8036C3.65569 10.7589 3.72266 10.7321 3.80301 10.7232C3.88337 10.7054 3.95926 10.6964 4.03069 10.6964C4.10212 10.6964 4.18694 10.7009 4.28516 10.7098C4.38337 10.7187 4.44587 10.7232 4.47266 10.7232C5.24051 10.7232 5.84319 10.5089 6.28069 10.0804C6.71819 9.64286 6.93694 9.04018 6.93694 8.27232ZM11.0753 12.4509L11.6244 8.95536C11.6334 8.90179 11.62 8.85268 11.5843 8.80804C11.5485 8.76339 11.5039 8.74107 11.4503 8.74107H10.4325C10.3075 8.74107 10.2316 8.88839 10.2048 9.18304C9.96373 8.82589 9.53962 8.64732 8.93248 8.64732C8.28962 8.64732 7.74051 8.88839 7.28516 9.37054C6.83873 9.85268 6.61551 10.4196 6.61551 11.0714C6.61551 11.5982 6.7673 12.0179 7.07087 12.3304C7.38337 12.6429 7.79855 12.7991 8.31641 12.7991C8.56641 12.7991 8.82534 12.7455 9.09319 12.6384C9.36105 12.5312 9.57534 12.3884 9.73605 12.2098C9.70034 12.317 9.68248 12.4107 9.68248 12.4911C9.68248 12.6339 9.74051 12.7054 9.85659 12.7054H10.7807C10.9503 12.7054 11.0485 12.6205 11.0753 12.4509ZM17.0352 9.02232C17.053 9.00446 17.0619 8.97321 17.0619 8.92857C17.0619 8.88393 17.0441 8.84375 17.0084 8.80804C16.9727 8.76339 16.9325 8.74107 16.8878 8.74107H15.8566C15.7584 8.74107 15.678 8.78571 15.6155 8.875L14.1959 10.9643L13.6066 8.95536C13.5619 8.8125 13.4637 8.74107 13.3119 8.74107H12.3075C12.2628 8.74107 12.2227 8.76339 12.1869 8.80804C12.1512 8.84375 12.1334 8.88393 12.1334 8.92857C12.1334 8.94643 12.2182 9.20982 12.3878 9.71875C12.5664 10.2277 12.7539 10.7768 12.9503 11.3661C13.1557 11.9554 13.2628 12.2679 13.2718 12.3036C12.5396 13.3036 12.1736 13.8393 12.1736 13.9107C12.1736 14.0268 12.2316 14.0848 12.3477 14.0848H13.3789C13.4771 14.0848 13.5575 14.0402 13.62 13.9509L17.0352 9.02232ZM22.1512 8.27232C22.1512 7.74554 21.9771 7.36607 21.6289 7.13393C21.2896 6.89286 20.8432 6.77232 20.2896 6.77232H18.1602C17.9816 6.77232 17.8834 6.85714 17.8655 7.02678L16.995 12.4911C16.986 12.5446 16.9994 12.5937 17.0352 12.6384C17.0709 12.683 17.1155 12.7054 17.1691 12.7054H18.2673C18.3744 12.7054 18.4459 12.6473 18.4816 12.5312L18.7227 10.9777C18.7316 10.9062 18.7628 10.8482 18.8164 10.8036C18.87 10.7589 18.9369 10.7321 19.0173 10.7232C19.0977 10.7054 19.1735 10.6964 19.245 10.6964C19.3164 10.6964 19.4012 10.7009 19.4994 10.7098C19.5977 10.7187 19.6602 10.7232 19.6869 10.7232C20.4548 10.7232 21.0575 10.5089 21.495 10.0804C21.9325 9.64286 22.1512 9.04018 22.1512 8.27232ZM26.2896 12.4509L26.8387 8.95536C26.8477 8.90179 26.8343 8.85268 26.7986 8.80804C26.7628 8.76339 26.7182 8.74107 26.6646 8.74107H25.6468C25.5218 8.74107 25.4459 8.88839 25.4191 9.18304C25.1869 8.82589 24.7628 8.64732 24.1468 8.64732C23.5039 8.64732 22.9548 8.88839 22.4994 9.37054C22.053 9.85268 21.8298 10.4196 21.8298 11.0714C21.8298 11.5982 21.9816 12.0179 22.2852 12.3304C22.5977 12.6429 23.0128 12.7991 23.5307 12.7991C23.7896 12.7991 24.053 12.7455 24.3209 12.6384C24.5887 12.5312 24.7985 12.3884 24.9503 12.2098C24.9503 12.2187 24.9414 12.2589 24.9236 12.3304C24.9057 12.4018 24.8968 12.4554 24.8968 12.4911C24.8968 12.6339 24.9548 12.7054 25.0709 12.7054H25.995C26.1646 12.7054 26.2628 12.6205 26.2896 12.4509ZM29.2093 6.97321V6.95982C29.2093 6.83482 29.1512 6.77232 29.0352 6.77232H28.0441C27.9459 6.77232 27.8878 6.82143 27.87 6.91964L26.9994 12.4911L26.9861 12.5179C26.9861 12.5625 27.0039 12.6071 27.0396 12.6518C27.0753 12.6875 27.12 12.7054 27.1736 12.7054H28.0575C28.2271 12.7054 28.3209 12.6205 28.3387 12.4509L29.2093 6.97321ZM5.31641 8.76786C5.27176 9.08036 5.15569 9.28571 4.96819 9.38393C4.78069 9.48214 4.51284 9.53125 4.16462 9.53125L3.72266 9.54464L3.95034 8.11161C3.96819 8.01339 4.02623 7.96428 4.12444 7.96428H4.37891C4.73605 7.96428 4.99498 8.01786 5.15569 8.125C5.31641 8.22321 5.36998 8.4375 5.31641 8.76786ZM30.9236 1.85714V19C30.9236 19.4643 30.7539 19.8661 30.4146 20.2054C30.0753 20.5446 29.6736 20.7143 29.2093 20.7143H1.78069C1.31641 20.7143 0.914621 20.5446 0.575335 20.2054C0.236049 19.8661 0.0664062 19.4643 0.0664062 19V1.85714C0.0664062 1.39286 0.236049 0.991071 0.575335 0.651785C0.914621 0.312499 1.31641 0.142856 1.78069 0.142856H29.2093C29.6736 0.142856 30.0753 0.312499 30.4146 0.651785C30.7539 0.991071 30.9236 1.39286 30.9236 1.85714Z" fill="#495057"/>

</svg>

</div>

</div>

</div>

</div>

</>

);

};

import React, { useState, useEffect } from 'react';

import { loginUser } from '../../../FrontendRequests/Requests-Api/User';

export default function Loginformular() {

const [email, setEmail] = useState('');

const [password, setPassword] = useState('');

const [isButtonDisabled, setIsButtonDisabled] = useState(true);

const [errorMessage, setErrorMessage] = useState('');

useEffect(() => {

// Check fields validity on input change

const isValid = password.trim() !== '' && email.trim() !== '' && isValidEmail(email.trim());

setIsButtonDisabled(!isValid);

}, [email, password]);

// Validate email format

const isValidEmail = (email: string) => {

const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

return emailRegex.test(email);

};

const handleLogin = async (event: React.FormEvent) => {

event.preventDefault();

if (password.trim() !== '' && email.trim() !== '') {

const result = await loginUser(email.trim(), password.trim());

if (result) {

window.location.href = '/';

} else {

setErrorMessage('Login ikke korrekt. Prøv igen');

}

} else {

setErrorMessage('Felterne skal udfyldes korrekt');

}

};

return (

<>

<section className="bg-gray-100 d-flex justify-content-center align-items-center">

<div className="row w-100">

<div className="col-12 col-lg-6 p-3 p-md-5 mx-auto">

<form onSubmit={handleLogin}>

<div className="form-group">

<label>Email</label>

<input

id="emailbox"

type="email"

className="form-control"

placeholder="Indtast din email addresse"

value={email}

onChange={(e) => setEmail(e.target.value)}

/>

</div>

<div className="form-group">

<label>Kodeord</label>

<input

id="password"

type="password"

className="form-control"

placeholder="Enter din kodeord"

value={password}

onChange={(e) => setPassword(e.target.value)}

/>

</div>

{errorMessage && <p className="text-danger">{errorMessage}</p>}

<button

type="submit"

className="btn btn-dark w-100 mt-4"

id="loginbutton"

disabled={isButtonDisabled}

>

Login

</button>

</form>

<div className="text-center">

<p>

Intet brugernavn: <a href="Register" style={{ color: 'blue' }}>Register</a>

</p>

</div>

</div>

</div>

</section>

</>

);

}

import React, { useEffect } from "react";

const Logudformular = () => {

useEffect(() => {

const timer = setTimeout(() => {

localStorage.removeItem("jwtToken")

window.location.href = '/';

}, 3000);

return () => clearTimeout(timer); // Cleanup the timer

}, []);

return (

<div className='d-flex justify-content-center'>

<p>You will be redirected to the main page in 3 seconds...</p>

</div>

);

};

export default Logudformular;

import OrderCardProduct from './orderCardProduct';

import OrderSummary from '../cart/orderSummary';

import React, {useEffect, useState } from 'react';

import { FetchOrderById } from '../../../FrontendRequests/Requests-Api/Order';

import type { Order, OrdersProduct } from '../../../FrontendRequests/Requests-Api/Order';

import { GetProduct } from '../../../FrontendRequests/Requests-Api/Product';

import type { Product } from '../../../FrontendRequests/Requests-Api/Product';

import { getShippingPrice } from '../../../FrontendRequests/Enums/shippingprice';

import { Button, Form, Modal } from 'react-bootstrap';

import { generateOrderConfirmationPDF, type OrderDetails } from '../FileGenerator/OrderSummery';

import { SendMesssage, SendOrderEmail } from '../../../FrontendRequests/Requests-Api/Email';

import type { SendOrderParams } from '../../../FrontendRequests/Requests-Api/Email';

import { Verifytoken } from '../../../FrontendRequests/Requests-Api/security';

import Editorder from './editorder';

import type { ProductImage } from '../../../FrontendRequests/Requests-Api/ProductImages';

interface Props {

order: Order | null,

isadmin: boolean

}

export default function OrderSummaries({ order, isadmin }: Props) {

const [Order, SetOrder] = useState<Order | null>(null);

const [SubTotal, SetSubTotal] = useState<number>(0);

const [Moms, SetMoms] = useState<number>(0);

const [showModal, setShowModal] = useState(false);

const [showTrackingNumberModal, setTrackingNumberModal] = useState(false);

const [showStatusModal, setshowStatusModal] = useState(false);

const [trackingnumber, Settrackingnumber] = useState<string>();

const [deliveryservice, Setdeliveryservice] = useState<string>();

function uint8ArrayToBase64(uint8Array: Uint8Array): string {

// Convert the Uint8Array to a regular array of numbers

const binaryString = String.fromCharCode.apply(

null,

Array.from(uint8Array) // Convert Uint8Array to regular array

);

return btoa(binaryString); // Convert binary string to base64

}

async function handleSubmitTrackingNumber() {

// Check if trackingnumber and deliveryservice are not empty

if (!trackingnumber || !deliveryservice) {

alert("Tracking nummer eller levering-websted må ikke være tom.");

} else {

const message = `Din Tracking nummer: ${trackingnumber}

Brug Tracking nummeret til at spore din ordre på: ${deliveryservice}

Du kan altid kontakte os på vores hjemmeside: ${import.meta.env.STORENAME} Hvis du har brug for hjælp! `;

const subject = `${import.meta.env.STORENAME} - Din Tracking nummer`;

const messageresult = await SendMesssage(order?.email as string, message, subject);

if (messageresult.result) {

alert("Tracking nummer er sendt");

setTrackingNumberModal(false);

} else {

alert("Tracking nummer kunne ikke sendes");

setTrackingNumberModal(false);

}

}

}

async function SendOrderConfirmation(order : Order) {

const sampleStatement: OrderDetails = {

companyname: import.meta.env.STORENAME,

companyaddress: import.meta.env.STOREADDRESS,

companyemail: import.meta.env.STOREEMAIL,

companyphone: import.meta.env.STOREPHONE,

orderid: order?.orderid.toString() as string,

useremail: order?.email as string,

invoiceDate: new Date()

.toLocaleDateString('en-GB')

.split('/')

.join('/'),

billingDetails: `${order?.firstname} ${order?.lastname} \n \n ${order?.email}`,

shippingDetails: `${order?.address}`,

comments: "Betaling forfalder inden for 30 dage. Vær venlig at angive Ordrenr i din betalingsmetode",

orderProducts: order?.order\_products as OrdersProduct[],

subtotal: order?.totalprice as number,

salesTax: order?.order\_products?.reduce((acc, product) =>

acc + parseFloat((Number(product.udgående\_moms)).toString()) \* product.quantity, 0

) as number,

shippingprice: getShippingPrice(order?.order\_products.length)

};

generateOrderConfirmationPDF(sampleStatement).then((pdfbytes) => {

const sendorder : SendOrderParams = {

email: order?.email as string,

pdfbase64: uint8ArrayToBase64(pdfbytes),

Filename: "ordrebekræftelse.pdf"

}

SendOrderEmail(sendorder).then((result) => {

})

})

}

const getorder = async (order: string | Order) => {

try {

if (typeof order === 'string') {

const result = await FetchOrderById(order)

if (result.result) {

const Products = await getproducts(result.data)

if (Products) {

result.data.products = Products

SetOrder(result.data);

SetSubTotal(result.data.totalprice);

SetMoms(result.data.order\_products?.reduce((acc: number, product: { udgående\_moms: any; quantity: number; }) =>

acc + parseFloat((Number(product.udgående\_moms)).toString()) \* product.quantity, 0

) as number)

SendOrderConfirmation(result.data as Order);

} else {

}

} else {

}

} else {

const Products = await getproducts(order)

if (Products) {

order.products = Products

SetOrder(order);

SetSubTotal(order.totalprice);

SetMoms(order.order\_products?.reduce((acc: number, product: { udgående\_moms: any; quantity: number; }) =>

acc + parseFloat((Number(product.udgående\_moms)).toString()) \* product.quantity, 0

) as number)

} else {

}

}

} catch (error) {

console.error('Error Getting products:', error);

}

};

const getproducts = async (Order: Order) => {

const updatedOrderProducts = await Promise.all(

Order.order\_products.map(async (orderProduct) => {

const product = await GetProduct(orderProduct.productid.toString());

if (!product.result) {

return {

productid: orderProduct.productid,

name: orderProduct.productname,

description: "",

features: "",

link: "",

købspris\_ex\_moms: orderProduct.købspris\_ex\_moms,

salgpris\_ex\_moms: orderProduct.salgpris\_ex\_moms,

indgående\_moms: orderProduct.indgående\_moms,

udgående\_moms: orderProduct.udgående\_moms,

tags: "",

barcode: 32342334,

images: [

{

imageid: 1,

productid: 0,

image\_url: "",

description: "",

created\_at: new Date().toISOString(),

},

] as ProductImage[],

productcolorsizes: [],

manufacturer: "",

categories: [],

} as Product;

}

(product.data as Product).købspris\_ex\_moms = orderProduct.købspris\_ex\_moms;

(product.data as Product).salgpris\_ex\_moms = orderProduct.salgpris\_ex\_moms;

(product.data as Product).indgående\_moms = orderProduct.indgående\_moms;

(product.data as Product).udgående\_moms = orderProduct.udgående\_moms;

return { ...product.data };

})

);

return updatedOrderProducts;

};

useEffect(() => {

if (order == null) {

const urlSearchParams = new URLSearchParams(window.location.search);

const params = Object.fromEntries(urlSearchParams.entries());

if (params.authtoken !== null) {

Verifytoken(params.authtoken).then((result) => {

if (params.orderid !== null && result.result) {

getorder(params.orderid)

setShowModal(true);

const timer = setTimeout(() => {

setShowModal(false);

}, 5000);

return () => clearTimeout(timer); // Cleanup the timer on unmount

}

})

} else {

}

} else {

getorder(order)

}

}, []);

return (

<>

<div className="p-3 p-md-5 bg-gray-100 rounded-2">

<div className="d-block d-md-flex justify-content-between align-items-start mb-4">

<div>

<h3 className="mb-2">Order #{Order?.orderid}</h3>

<p className="mb-0 text-dark">Total Products: <b>{Order?.order\_products?.length}</b></p>

</div>

<div style={{ display: 'flex', gap: '10px' }}>

{isadmin ? (

<Button

className="custom-dropdown-toggle text\_style"

variant="warning"

onClick={() => {

const sampleStatement: OrderDetails = {

companyname: import.meta.env.STORENAME,

companyaddress: import.meta.env.STOREADDRESS,

companyemail: import.meta.env.STOREEMAIL,

companyphone: import.meta.env.STOREPHONE,

orderid: order?.orderid.toString() as string,

useremail: order?.email as string,

invoiceDate: new Date().toISOString().split('T')[0],

billingDetails: `${order?.firstname} ${order?.lastname} \n \n ${order?.email}`,

shippingDetails: `${order?.address}`,

comments: "Betaling forfalder inden for 30 dage. Vær venlig at angive Ordrenr i din betalingsmetode",

orderProducts: order?.order\_products as OrdersProduct[],

subtotal: order?.totalprice as number,

salesTax: order?.order\_products?.reduce((acc, product) =>

acc + parseFloat((Number(product.udgående\_moms)).toString()) \* product.quantity, 0

) as number,

shippingprice: getShippingPrice(order?.order\_products.length)

};

generateOrderConfirmationPDF(sampleStatement).then((pdfbytes) => {

const blob = new Blob([pdfbytes], { type: 'application/pdf' });

const link = document.createElement('a');

link.href = URL.createObjectURL(blob);

link.download = 'ordrebekræftelse.pdf';

link.click();

console.log('Rapport hentet!');

});

}}

>

Download PDF

</Button>

) : null}

{isadmin ? (

<div>

<Button

className="custom-dropdown-toggle text\_style"

variant="primary"

onClick={() => {

setTrackingNumberModal(true)

}}

>

Send tracking nummer

</Button>

</div>

) : null}

{isadmin ? (

<div>

<Button

className="custom-dropdown-toggle text\_style"

style={{ backgroundColor: 'purple', borderColor: 'purple' }}

variant="primary"

onClick={() => {

setshowStatusModal(true);

}}

>

Ændre Status

</Button>

</div>

) : null}

</div>

</div>

{Order?.products?.map((product, index) => (

<OrderCardProduct

key={index} // Use a unique key, if available, such as product.id

product={product}

status={Order.statusid}

quantity={Order.order\_products[index].quantity}

address={`${Order.address}`}

email={Order.email}

phoneNumber={Order.phonenumber as number}

/>

))}

<div className="card shadow-xs border p-3 p-md-4 mb-4">

<div className="row">

<div className="col-12 col-lg-3 mt-4 mt-md-0">

<h5 className="text-base">Billing Address</h5>

<p className="text-sm mb-0 opacity-8 pe-md-7">{Order?.address}</p>

</div>

<div className="col-12 col-lg-3 mt-4 mt-lg-0">

<h5 className="text-base">Payment information</h5>

<div className="d-flex">

<svg width="49" height="32" viewBox="0 0 49 32" fill="none" xmlns="http://www.w3.org/2000/svg">

<g clip-path="url(#clip0\_152\_7843)">

<path d="M22.1996 20.8737H19.6074L21.2288 10.8484H23.8209L22.1996 20.8737Z" fill="#00579F"/>

<path d="M31.5982 11.0936C31.0869 10.8907 30.276 10.6667 29.2732 10.6667C26.7133 10.6667 24.9106 12.0318 24.8996 13.9835C24.8783 15.4234 26.1902 16.2232 27.1714 16.7033C28.1743 17.1939 28.5153 17.5141 28.5153 17.9514C28.5051 18.623 27.7049 18.9326 26.9585 18.9326C25.9236 18.9326 25.3691 18.773 24.5263 18.3993L24.185 18.2392L23.8223 20.4896C24.4302 20.7666 25.5504 21.0124 26.7133 21.0232C29.4332 21.0232 31.204 19.6792 31.225 17.5994C31.2353 16.4582 30.5426 15.5837 29.0491 14.8691C28.1424 14.4104 27.5872 14.1011 27.5872 13.6318C27.5978 13.2051 28.0568 12.7681 29.0803 12.7681C29.923 12.7467 30.5422 12.9492 31.0112 13.1519L31.2457 13.2584L31.5982 11.0936Z" fill="#00579F"/>

<path d="M35.0438 17.3221C35.2573 16.7461 36.0787 14.5171 36.0787 14.5171C36.068 14.5385 36.2918 13.9305 36.4198 13.5573L36.601 14.4211C36.601 14.4211 37.0918 16.8208 37.1984 17.3221C36.7933 17.3221 35.5558 17.3221 35.0438 17.3221ZM38.2436 10.8484H36.2385C35.6202 10.8484 35.1504 11.0295 34.8836 11.6802L31.0332 20.8735H33.7531C33.7531 20.8735 34.2009 19.6362 34.2972 19.3697C34.5956 19.3697 37.2415 19.3697 37.6254 19.3697C37.6998 19.7217 37.9347 20.8735 37.9347 20.8735H40.3348L38.2436 10.8484Z" fill="#00579F"/>

<path d="M17.4429 10.8484L14.9043 17.6847L14.6269 16.2982C14.1575 14.6984 12.6856 12.9602 11.043 12.096L13.3683 20.863H16.1095L20.184 10.8484H17.4429Z" fill="#00579F"/>

<path d="M12.5473 10.8484H8.37665L8.33398 11.051C11.5873 11.8829 13.742 13.8883 14.6272 16.2986L13.7206 11.6911C13.5713 11.0508 13.1126 10.8695 12.5473 10.8484Z" fill="#FAA61A"/>

</g>

<rect x="0.833984" y="0.5" width="47" height="31" rx="5.5" stroke="#DDE0E5"/>

<defs>

<clipPath id="clip0\_152\_7843">

<rect width="32" height="32" fill="white" transform="translate(8.33398)"/>

</clipPath>

</defs>

</svg>

<p className="text-sm ms-3">

<b>{Order?.paymentname}</b> <br/>

</p>

</div>

</div>

<div className="col-12 col-lg-6">

<OrderSummary

subtotal={SubTotal}

shippingprice={getShippingPrice(Order?.order\_products.length)}

moms={Moms}

/>

</div>

</div>

</div>

{/\* Modal for Order \*/}

<Modal show={showModal} size="lg" centered>

<Modal.Header closeButton className="bg-success text-white">

<Modal.Title className="d-flex align-items-center">

<i className="mdi mdi-check-circle-outline me-2" style={{ fontSize: "1.5rem" }}></i>

Ordre gennemført med succes

</Modal.Title>

</Modal.Header>

<Modal.Body className="text-center p-4">

<i className="mdi mdi-cart-check-outline" style={{ fontSize: "4rem", backgroundColor: "lightgreen" }}></i>

<p className="mt-3 fs-5">

Tak for dit køb! Din ordre er blevet registreret og er under behandling

</p>

<p className="mt-3 fs-5">

Du får en ordrebekræftelse i din indbakke snarest - Tjek din spam eller uønsket email

</p>

<p className="mt-3 fs-5">

Du får en tracking nummer så du kan spore din ordre så snart den er færdig behandlet! - Tjek din spam eller uønsket email

</p>

</Modal.Body>

<Modal.Footer className="d-flex justify-content-center">

</Modal.Footer>

</Modal>

{/\* Modal for Sending tracking number \*/}

<Modal show={showTrackingNumberModal} onHide={() => setTrackingNumberModal(false)} size="lg" centered>

<Modal.Header closeButton className="text-white" style={{ backgroundColor: "blue" }}>

<Modal.Title className="d-flex align-items-center">

<i className="mdi mdi-package-check me-2" style={{ fontSize: "1.5rem" }}></i>

Send tracking number

</Modal.Title>

</Modal.Header>

<Modal.Body>

{/\* Textfield for the tracking number \*/}

<Form>

<Form.Group controlId="trackingNumber">

<Form.Label>Tracking nummer</Form.Label>

<Form.Control

type="text"

placeholder="Indtast tracking nummer"

value={trackingnumber}

onChange={(e) => Settrackingnumber(e.target.value)}

/>

</Form.Group>

</Form>

{/\* Textfield for delivery service \*/}

<Form>

<Form.Group controlId="deliveryService">

<Form.Label>Leverings-websted</Form.Label>

<Form.Control

type="text"

placeholder="eksempel: GLS.com"

value={deliveryservice}

onChange={(e) => Setdeliveryservice(e.target.value)}

/>

</Form.Group>

</Form>

</Modal.Body>

<Modal.Footer>

<Button variant="secondary" onClick={() => setTrackingNumberModal(false)}>

Close

</Button>

<Button variant="primary" onClick={handleSubmitTrackingNumber}>

Submit

</Button>

</Modal.Footer>

</Modal>

{/\* Modal for Changing status \*/}

<Modal show={showStatusModal} onHide={() => setshowStatusModal(false)} size="lg" centered>

<Modal.Header closeButton className="text-white" style={{ backgroundColor: "purple" }}>

<Modal.Title className="w-100 d-flex justify-content-center">

<Editorder order={Order as Order} />

</Modal.Title>

</Modal.Header>

</Modal>

</div>

</>

);

};

import React, { useState, type JSX } from 'react';

import { Container, Row, Col, Card } from 'react-bootstrap';

const Privatpolitiks = () => {

const [openSection, setOpenSection] = useState<string | null>(null);

const handleToggle = (section: string) => {

setOpenSection(openSection === section ? null : section);

};

const renderSection = (section: string, title: string, content: string | JSX.Element) => {

return (

<Card className="mb-4">

<Card.Header as="h5" onClick={() => handleToggle(section)} style={{ cursor: 'pointer' }}>

{title}

</Card.Header>

{openSection === section && (

<Card.Body>

{/\* Check if content is a string or JSX.Element \*/}

{typeof content === 'string' ? (

<p dangerouslySetInnerHTML={{ \_\_html: content }} />

) : (

content

)}

</Card.Body>

)}

</Card>

);

};

return (

<Container className="my-5">

<Row className="text-center mb-4">

<Col>

<h1>Privatlivspolitik for {import.meta.env.COMPANYNAME}</h1>

</Col>

</Row>

{renderSection(

'section1',

'Introduktion',

`Hos ${import.meta.env.STORENAME} tager vi beskyttelsen af dine personlige oplysninger meget alvorligt. Denne

privatlivspolitik beskriver, hvordan vi indsamler, bruger, opbevarer og beskytter dine personlige data,

når du interagerer med vores hjemmeside, kontakter os, eller benytter vores bygge- og murertjenester. Vi

sikrer, at dine personlige data behandles i overensstemmelse med gældende lovgivning, herunder EU's General

Data Protection Regulation (GDPR).`

)}

{renderSection(

'section2',

'Hvilke personoplysninger indsamler vi?',

`Vi indsamler personoplysninger som navn, adresse, telefonnummer, e-mailadresse, personnummer (ved behov),

oplysninger om byggeprojekter og betalingsoplysninger for at kunne tilbyde vores tjenester.`

)}

{renderSection(

'section3',

'Hvordan indsamler vi dine personoplysninger?',

`Dine personoplysninger indsamles direkte fra dig, automatisk via vores hjemmeside (f.eks. cookies og

IP-adresser), samt fra tredjeparter som samarbejdspartnere eller sociale platforme.`

)}

{renderSection(

'section4',

'Hvorfor indsamler vi dine personoplysninger?',

`Vi bruger dine oplysninger til at opfylde kontraktlige forpligtelser, kommunikere med dig, overholde

lovgivningen og forbedre vores service.`

)}

{renderSection(

'section5',

'Hvem deler vi dine personoplysninger med?',

`Vi deler kun dine data med vores ansatte, samarbejdspartnere, leverandører, offentlige myndigheder (hvis

nødvendigt) og forsikringsselskaber ved skader eller forsikringskrav. Vi sælger aldrig dine personoplysninger

til tredjepart.`

)}

{renderSection(

'section6',

'Hvordan beskytter vi dine personoplysninger?',

`Vi beskytter dine oplysninger med sikre servere, kryptering, begrænset adgang for medarbejdere og regelmæssige

sikkerhedsvurderinger.`

)}

{renderSection(

'section7',

'Hvor længe opbevarer vi dine personoplysninger?',

`Vi opbevarer dine data, så længe det er nødvendigt for formålet med indsamlingen og for at overholde

lovgivningen. Når dataene ikke længere er nødvendige, sletter vi dem.`

)}

{renderSection(

'section8',

'Dine rettigheder',

`Du har ret til indsigt i, berigtigelse af, sletning af og begrænsning af behandlingen af dine personoplysninger.

Du har også ret til dataportabilitet og indsigelse.`

)}

{renderSection(

'section9',

'Cookies og sporingsteknologier',

`Vores hjemmeside bruger cookies for at forbedre din oplevelse og levere relevante tjenester. Vi anvender

nødvendige cookies, præference-cookies, statistik-cookies og marketing-cookies. Du kan administrere eller

slette cookies via Cookie-indstillinger i bunden af siden.`

)}

{renderSection(

'section10',

'Ændringer i privatlivspolitikken',

`Vi forbeholder os retten til at opdatere denne politik. Eventuelle ændringer offentliggøres på vores hjemmeside.`

)}

{renderSection(

'section11',

'Kontaktoplysninger',

`Har du spørgsmål, kontakt os venligst på: <br />

<strong>${import.meta.env.STORENAME}</strong>

<br />

<strong>${import.meta.env.STOREADDRESS}</strong>

<br />

<strong>${import.meta.env.STOREPHONE}</strong>

<br />

<strong>${import.meta.env.STOREEMAIL}</strong>`

)}

{renderSection(

'section12',

'Returret',

`I henhold til dansk lovgivning har du som forbruger 14 dages fortrydelsesret ved køb af varer online. Hvis du ønsker at returnere en vare, skal den være i ubrugt og ubeskadiget stand. Kontakt os inden for de 14 dage for at arrangere en returnering. Fragtomkostninger i forbindelse med returnering afholdes af kunden.`

)}

{renderSection(

'section13',

'Reklamation',

`Som forbruger har du ifølge dansk købelov en reklamationsret på 2 år. Det betyder, at du kan få en defekt vare repareret, ombyttet eller få pengene tilbage afhængigt af situationen. Hvis du opdager en fejl eller mangel ved din vare, bedes du kontakte os hurtigst muligt med dokumentation for fejlen.`

)}

</Container>

);

};

export default Privatpolitiks;

import React, { useEffect, useState } from 'react';

import ProductBadge from './productBadge';

import ProductImages from './productImages';

import ProductAccordion from './productAccordion';

import { GetProduct, type Product, type ProductColorSize } from '../../../FrontendRequests/Requests-Api/Product';

import { useNotifications } from '../NotifcationSystem/NotificationContext';

import { Form } from 'react-bootstrap';

export default function ProductOverviewGallery() {

const { addToBasket } = useNotifications();

const [GetProductState, SetProductState] = useState<Product | null>(null);

const [GetIdentifier, SetIdentifier] = useState<ProductColorSize | null>(null);

const [defaultproduct, setdefaultproduct] = useState<boolean>(false);

const [uniquecolors, setuniquecolors] = useState<ProductColorSize[]>();

const [uniquesizes, setuniquesizes] = useState<ProductColorSize[]>();

const [selectedcolor, setselectedcolor] = useState<number>();

const [selectedsize, setselectedsize] = useState<number>();

function isProductColorSize(obj: any): obj is ProductColorSize {

return (

typeof obj === 'object' &&

obj !== null &&

typeof obj.colorid === 'number' &&

typeof obj.colorname === 'string' &&

typeof obj.sizeid === 'number' &&

typeof obj.sizename === 'string' &&

typeof obj.quantity === 'number'

);

}

const handleColorSelect = (e: { target: { value: any; }; }) => {

const sizes = GetProductState?.productcolorsizes.filter((colors) => colors?.colorid === Number(e.target.value)) as ProductColorSize[]

const uniquesizes = sizes

.filter((item, index, self) =>

index === self.findIndex((t) => (

t.sizename === item.sizename // Check for unique size names

))

)

setuniquesizes(uniquesizes)

setselectedcolor(Number(e.target.value))

};

const handlesizeSelect = (e: { target: { value: any; }; }) => {

setselectedsize(Number(e.target.value))

};

const Getproductfunc = async (productid: string, Identifier: ProductColorSize) => {

try {

if (productid !== null) {

const result = await GetProduct(productid);

if (result.result) {

const product = (result.data as Product)

product.productcolorsizes = product.productcolorsizes.filter(item => item.quantity > 0);

SetProductState(product);

if (isProductColorSize(Identifier)) {

SetIdentifier(Identifier);

const productcolorsize = Identifier as ProductColorSize

if (productcolorsize.colorname === "Default") {

setdefaultproduct(true)

setselectedcolor(productcolorsize.colorid)

setselectedsize(productcolorsize.sizeid)

}

}

}

}

} catch (error) {

console.error('Error Getting products:', error);

}

};

useEffect(() => {

const urlSearchParams = new URLSearchParams(window.location.search);

const params = Object.fromEntries(urlSearchParams.entries());

if (params.id !== null) {

Getproductfunc(params.id, JSON.parse(params.Identifier));

}

}, []);

useEffect(() => {

const uniquecolors = GetProductState?.productcolorsizes

.filter((item, index, self) =>

index === self.findIndex((t) => (

t.colorname === item.colorname // Check for unique size names

))

)

const uniquesizes = GetProductState?.productcolorsizes

.filter((item, index, self) =>

index === self.findIndex((t) => (

t.sizename === item.sizename // Check for unique size names

))

)

setuniquecolors(uniquecolors)

setuniquesizes(uniquesizes)

}, [GetProductState]);

// Handler for "Add to Cart" button

const addToCartHandler = (productId:number) => {

if (selectedcolor && selectedsize) {

// add product to basket

const productcolorsize = GetProductState?.productcolorsizes.find(

(item) => item.colorid === selectedcolor && item.sizeid === selectedsize

)

if (productcolorsize !== undefined) {

addToBasket(productId, productcolorsize)

window.location.href = '/Butik'; // Redirects to the cart page with product id

} else {

alert("Kunne ikke finde produktet")

}

} else {

alert("Farve eller størrelse ikke valgt!")

}

};

return (

<>

<div className="card card-product card-plain">

<div className="row">

{GetProductState?.images.length !== 0 && GetProductState?.images != null && (

<ProductImages images={GetProductState?.images} />

)}

<div className="col-12 col-lg-6 mt-5 mt-lg-0">

{GetProductState?.name.length !== 0 && (

<h2 style={{ fontSize: '30px !important' }}>{GetProductState?.name}</h2>

)}

{GetProductState?.salgpris\_ex\_moms && (

<>

<div className="d-flex mb-3">

<h4 className="font-weight-bold">

{(Number(GetProductState?.salgpris\_ex\_moms) + Number(GetProductState?.udgående\_moms)).toLocaleString()} kr.

</h4>

<input className="opacity-0" defaultValue={(Number(GetProductState?.salgpris\_ex\_moms) + Number(GetProductState?.udgående\_moms)).toLocaleString()} />

</div>

</>

)}

<p style={{ fontSize: '15px !important' }} className="mt-4">

{GetProductState?.description}

</p>

{GetProductState?.productcolorsizes.length !== 0 && (

<>

<h6 className="mt-4">Farve:</h6>

{GetProductState?.productcolorsizes && (

<ProductBadge

addtocart={true}

productid={GetProductState.productid}

colorsize={GetProductState?.productcolorsizes}

/>

)}

</>

)}

<Form.Select className="input-select" onChange={handleColorSelect} defaultValue="Vælg Farver" disabled={defaultproduct}>

<option disabled>Vælg Farver</option>

{uniquecolors?.map((color) => (

<option value={color.colorid}>

{color.colorname}

</option>

))}

</Form.Select>

<h6 className="mt-4">Størrelser:</h6>

<Form.Select className="input-select" onChange={handlesizeSelect} defaultValue="Vælg Størrelser" disabled={defaultproduct}>

<option disabled>Vælg Størrelser</option>

{uniquesizes?.map((size) => (

<option value={size.sizeid}>

{size.sizename}

</option>

))}

</Form.Select>

<div className="d-flex align-items-center mt-4">

<button

className="btn btn-dark btn-lg mb-0 me-4"

onClick={() => {

if (!GetProductState && !GetIdentifier) {

return;

} else {

addToCartHandler(GetProductState?.productid as number);

}

}}

disabled={!GetProductState || !GetIdentifier} // Disable the button if either is falsy

>

Tilføj

</button>

<a href="#favorite">

<i id="heart1" className="far fa-heart text-2xl"></i>

</a>

</div>

{GetProductState && <ProductAccordion product={GetProductState} />}

</div>

</div>

</div>

</>

);

}

import React, { useState, useEffect } from 'react';

import { FetchAllCountry, type Country } from '../../../FrontendRequests/Requests-Api/Country';

import { RegisterUser, UserExsistByEmail } from "../../../FrontendRequests/Requests-Api/User";

import { CreateAddress } from "../../../FrontendRequests/Requests-Api/Address";

import { UserRoles } from "../../../FrontendRequests/Enums/Usertypes";

import { SendMesssage } from '../../../FrontendRequests/Requests-Api/Email';

export default function Registerformular() {

const [Countries, setCountries] = useState<Country[] | null>(null);

const [formValues, setFormValues] = useState({

user: '',

name: '',

surname: '',

email: '',

phone: '',

address: '',

city: '',

postalCode: '',

password: '',

confirmPassword: '',

ageCheck: false,

tradeCheck: false,

country: ''

});

const [isButtonDisabled, setIsButtonDisabled] = useState(true);

// Fetch countries

const GetCountries = async () => {

try {

const result = await FetchAllCountry();

setCountries(result.data);

} catch (error) {

console.error('Error fetching countries:', error);

setCountries([]);

}

};

useEffect(() => {

GetCountries();

}, []);

// Check form validity

useEffect(() => {

const isValid = Object.values(formValues).every(value =>

typeof value === 'string' ? value.trim() !== '' : true // Only trim if the value is a string

) && formValues.ageCheck && formValues.tradeCheck && formValues.country !== '';

setIsButtonDisabled(!isValid);

}, [formValues]);

// Handlers for form inputs

const handleInputChange = (e: React.ChangeEvent<HTMLInputElement | HTMLSelectElement>) => {

const { name, value, type } = e.target;

// Check if the target is an HTMLInputElement

if (type === 'checkbox') {

const { checked } = e.target as HTMLInputElement; // Type assertion to HTMLInputElement

setFormValues(prevState => ({

...prevState,

[name]: checked

}));

} else {

// For other input types, use value directly

setFormValues(prevState => ({

...prevState,

[name]: value

}));

}

};

// Validation functions

const postalCheck = (postalCode: string) => /^[0-9]+$/.test(postalCode);

const phoneCheck = (phone: string) => /^[0-9]{8}$/.test(phone);

const isValidEmail = (email: string) => /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email);

const passwordMatch = (password1: string, password2: string) => password1 === password2;

const passwordLength = (password: string) => password.length >= 10;

const nameCheck = (name: string) => /^[A-Za-z]{2,15}$/.test(name);

const surnameCheck = (surname: string) => /^[A-Za-z]{2,15}$/.test(surname);

const usernameCheck = (user: string) => /^[A-Za-z0-9\_]{5,10}$/.test(user);

const handleSubmit = async (event: React.FormEvent) => {

event.preventDefault();

if (postalCheck(formValues.postalCode) && phoneCheck(formValues.phone) && isValidEmail(formValues.email) &&

passwordMatch(formValues.password, formValues.confirmPassword) && passwordLength(formValues.password)

&& nameCheck(formValues.name) && surnameCheck(formValues.surname) && usernameCheck(formValues.user)) {

const userCheck = await UserExsistByEmail(formValues.email);

if (userCheck.result) {

alert("Email findes. Prøv igen");

} else {

const resultAddress = await CreateAddress(formValues.address, Number(formValues.postalCode), formValues.city, Number(formValues.country));

if (resultAddress.result) {

const addressId = resultAddress.data.addressid;

const resultRegister = await RegisterUser(

formValues.user, formValues.name, formValues.surname, formValues.password, formValues.email,

Number(formValues.phone), addressId, UserRoles.USER

);

if (resultRegister.result) {

const message = `Din bruger er nu oprettet på - ${import.meta.env.STORENAME} \n\n Brugernavn: ${resultRegister.data.username} \n Email: ${resultRegister.data.email} \n Kodeord: Kodeordet du oprettede \n\n Du kan anvende følgende oplysninger til at logge ind på ${import.meta.env.STORENAME} \n\n Venlig hilsen ${import.meta.env.STORENAME}`

await SendMesssage(formValues.email, message, `${import.meta.env.STORENAME} - Bruger oprettet`)

alert("Bruger oprettet");

window.location.href = '/';

} else {

alert("Bruger ikke oprettet");

}

} else {

alert("Addresse kunne ikke gemmes");

}

}

} else {

if (!postalCheck(formValues.postalCode)) alert("Postnummer skal være nummerisk");

if (!phoneCheck(formValues.phone)) alert("Telefon Nummer skal være på 8 tegn og være nummerisk");

if (!isValidEmail(formValues.email)) alert("EMAIL ER IKKE KORREKT");

if (!passwordMatch(formValues.password, formValues.confirmPassword)) alert("KODEORD MATCHER IKKE");

if (!passwordLength(formValues.password)) alert("KODEORD MÅ IKKE VÆRE MINDRE END 10 TEGN");

if (!nameCheck(formValues.name)) alert("Navn skal være mellem 2 og 15 bogstaver");

if (!surnameCheck(formValues.surname)) alert("Efternavn skal være mellem 2 og 15 bogstaver");

if (!usernameCheck(formValues.user)) alert("Brugernavn skal være mellem 5 og 15 bogstaver");

}

};

return (

<section className="bg-gray-100 d-flex justify-content-center align-items-center">

<div className="row w-100">

<div className="col-12 col-lg-6 p-3 p-md-5 mx-auto">

<form onSubmit={handleSubmit}>

<div className="form-group">

<label>Brugernavn</label>

<input

type="text"

className="form-control"

name="user"

value={formValues.user}

onChange={handleInputChange}

maxLength={10}

placeholder="Indtast din brugernavn"

/>

</div>

<div className="form-group">

<label>Fornavn</label>

<input

type="text"

className="form-control"

name="name"

value={formValues.name}

onChange={handleInputChange}

maxLength={15}

placeholder="Indtast dit fornavn"

/>

</div>

<div className="form-group">

<label>Efternavn</label>

<input

type="text"

className="form-control"

name="surname"

value={formValues.surname}

onChange={handleInputChange}

maxLength={15}

placeholder="Indtast dit Efternavn"

/>

</div>

<div className="form-group">

<label>Email</label>

<input

type="email"

className="form-control"

name="email"

value={formValues.email}

onChange={handleInputChange}

placeholder="Indtast din email addresse"

/>

</div>

<div className="form-group">

<label>Telefon Nummer</label>

<input

type="text"

className="form-control"

name="phone"

value={formValues.phone}

onChange={handleInputChange}

placeholder="Indtast din telefon nummer: 53948595"

/>

</div>

<div className="form-group">

<label>Gade/vej</label>

<input

type="text"

className="form-control"

name="address"

value={formValues.address}

onChange={handleInputChange}

placeholder="Indtast din addresse"

/>

</div>

<div className="form-group">

<label>By</label>

<input

type="text"

className="form-control"

name="city"

value={formValues.city}

onChange={handleInputChange}

placeholder="Indtast din by"

/>

</div>

<div className="form-group">

<label>Postnummer</label>

<input

type="text"

className="form-control"

name="postalCode"

value={formValues.postalCode}

onChange={handleInputChange}

maxLength={6}

placeholder="Indtast din Postnummer"

/>

</div>

<div className="form-group">

<label>Land</label>

<select

className="form-control"

name="country"

value={formValues.country}

onChange={handleInputChange}

>

<option value="" disabled>Vælg land</option>

{Countries?.map((country) => (

<option key={country.countryid} value={country.countryid}>

{country.country}

</option>

))}

</select>

</div>

<div className="form-group">

<label>Kodeord</label>

<input

type="password"

className="form-control"

name="password"

value={formValues.password}

onChange={handleInputChange}

placeholder="Indtast din kodeord"

/>

</div>

<div className="form-group">

<label>Gentag Kodeord</label>

<input

type="password"

className="form-control"

name="confirmPassword"

value={formValues.confirmPassword}

onChange={handleInputChange}

placeholder="Gentag din kodeord"

/>

</div>

<div className="d-flex align-items-center">

<div className="form-group form-check">

<input

type="checkbox"

className="form-check-input"

name="ageCheck"

checked={formValues.ageCheck}

onChange={handleInputChange}

/>

<label className="form-check-label" htmlFor="ageCheck">

Jeg er <strong>18+</strong>

</label>

</div>

<div className="form-group form-check ps-5">

<input

type="checkbox"

className="form-check-input"

name="tradeCheck"

checked={formValues.tradeCheck}

onChange={handleInputChange}

/>

<label className="form-check-label" htmlFor="tradeCheck">

Jeg accepter <strong>Handelsbetingelser</strong>

</label>

</div>

</div>

<button

className="btn btn-dark w-100 mt-4"

id="registerbutton"

type="submit"

disabled={isButtonDisabled}

>

Register

</button>

<div className="text-center">

<p>

Allerede bruger: <a href="Login" style={{ color: 'blue' }}>Login</a>

</p>

</div>

</form>

</div>

</div>

</section>

);

}

# **Database**

import pkg from 'pg';

const isProd = import.meta.env.PROD

var pool = {

connection: false,

pool: new pkg.Pool({})

}

if (isProd) {

try {

// Configure PostgreSQL connection

const dbcon = new pkg.Pool({

user: import.meta.env.USERCLOUD,

host: import.meta.env.HOSTCLOUD,

database: import.meta.env.DATABASE\_NAMECLOUD,

password: String(import.meta.env.PASSWORDCLOUD),

port: Number(import.meta.env.DATABASEPORTCLOUD),

ssl: {

rejectUnauthorized: false, // Render requires SSL, disable strict SSL verification

},

});

await dbcon.connect(); // Attempt to connect

console.log('Connected to PostgreSQL successfully!');

console.log('ENV: CLOUD');

console.log(`Databasename : ${import.meta.env.DATABASE\_NAMECLOUD}`);

pool.connection = true;

pool.pool = dbcon;

} catch (error) {

console.error('Error connecting to PostgreSQL:', error.message);

pool.connection = false;

pool.pool = new pkg.Pool({});

} finally {

}

} else {

try {

// Configure PostgreSQL connection

const dbcon = new pkg.Pool({

user: import.meta.env.USER,

host: import.meta.env.HOST,

database: import.meta.env.DATABASE\_NAME,

password: String(import.meta.env.PASSWORD),

port: Number(import.meta.env.DATABASEPORT),

});

await dbcon.connect(); // Attempt to connect

console.log('Connected to PostgreSQL successfully!');

console.log('ENV: LOCAL');

console.log(`Databasename : ${import.meta.env.DATABASE\_NAME}`);

pool.connection = true;

pool.pool = dbcon;

} catch (error) {

console.error('Error connecting to PostgreSQL:', error.message);

pool.connection = false;

pool.pool = new pkg.Pool({});

} finally {

}

}

export default pool;

export interface Address {

addressid: number;

address: string;

postalcode: number;

city: string;

countryid: number;

}

interface ApiResponse {

result: boolean,

message?: string;

data?: any;

}

export async function CreateAddress(address: string, postalcode: number, city: string, countryid: number): Promise<ApiResponse> {

try {

const response = await fetch(`/api/address/CreateAddress`, {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify({ address, postalcode, city, countryid }),

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData, data: null };

}

const data = await response.json();

return { result: true, message: '', data: data };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: "", data: null };

}

}

export async function UpdateAddress(

id: number,

address: string,

postalcode: number,

city: string,

countryid: number

): Promise<ApiResponse> {

try {

const response = await fetch(`/api/address/UpdateAddress?id=${id}`, {

method: 'PUT',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify({ address, postalcode, city, countryid }),

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData.error, data: null };

}

const data = await response.json();

return { result: true, message: 'Address updated successfully', data };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: 'Request failed', data: null };

}

}

export interface Category {

categoryid: number;

categoryname: string;

categoryimage: string

categorydescription: string;

}

interface ApiResponse {

result: boolean,

message?: string;

data?: any;

}

export async function FetchAllCateGories(): Promise<ApiResponse> {

try {

const response = await fetch(`/api/category/GetCategories`, {

method: 'GET',

headers: {

'Content-Type': 'application/json'

},

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData, data: [] };

}

const countries: Category[] = await response.json();

return { result: true, message: 'Categories fetched', data: countries};

} catch (error) {

console.error('Categories failed:', error);

return { result: false, message: "", data: null };

}

}

export async function AddCategorY(

categoryname: string,

categoryimage: string,

categorydescription: string

): Promise<ApiResponse> {

try {

const response = await fetch(`/api/category/CreateCategory`, {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({

categoryname,

categoryimage,

categorydescription,

}),

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error adding category:', errorData.error);

return { result: false, message: errorData.error, data: null };

}

const newCategory: Category = await response.json();

return { result: true, message: 'Category added successfully', data: newCategory };

} catch (error) {

console.error('Failed to add category:', error);

return { result: false, message: "", data: null };

}

}

export async function GetCategoryById(categoryid: string): Promise<ApiResponse> {

try {

const response = await fetch(`/api/category/GetCategoryById?id=${categoryid}`, {

method: 'GET',

headers: {

'Content-Type': 'application/json',

},

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error fetching category:', errorData.error);

return { result: false, message: errorData.error, data: null };

}

const category: Category = await response.json();

return { result: true, message: 'Category fetched successfully', data: category };

} catch (error) {

console.error('Failed to fetch category:', error);

return { result: false, message: "", data: null };

}

}

export async function UpdateCategory(

categoryid: number,

categoryname?: string,

categoryimage?: string,

categorydescription?: string

): Promise<ApiResponse> {

try {

// Send the PUT request with the update data

const response = await fetch(`/api/category/UpdateCategory?id=${categoryid}`, {

method: 'PUT',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({

categoryname,

categoryimage,

categorydescription,

}),

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error updating category:', errorData.error);

return { result: false, message: errorData.error, data: null };

}

const updatedCategory: Category = await response.json();

return { result: true, message: 'Category updated successfully', data: updatedCategory };

} catch (error) {

console.error('Failed to update category:', error);

return { result: false, message: "", data: null };

}

}

export async function DeleteCategoryById(categoryid: number): Promise<ApiResponse> {

try {

const response = await fetch(`/api/category/DeleteCategory?id=${categoryid}`, {

method: 'DELETE',

headers: {

'Content-Type': 'application/json',

},

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error deleting category:', errorData.error);

return { result: false, message: 'Category not found', data: null };

}

// Category deleted successfully, no content to return

const addedcategory: Category = await response.json();

return { result: true, message: 'Category deleted successfully', data: addedcategory };

} catch (error) {

console.error('Failed to delete category:', error);

return { result: false, message: "", data: null };

}

}

export interface Color {

colorid: number;

colorname: string;

}

interface ApiResponse {

result: boolean,

message: string;

data?: any;

}

export async function GetAllColors(): Promise<ApiResponse> {

try {

const response = await fetch(`/api/color/GetColors`, {

method: 'GET',

headers: {

'Content-Type': 'application/json'

},

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData, data: [] };

}

const colors: Color[] = await response.json();

return { result: true, message: 'Colors fetched', data: colors };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: "", data: null };

}

}

export async function CreateColor(colorname: string): Promise<ApiResponse> {

try {

const response = await fetch(`/api/color/CreateColor`, {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify({ colorname })

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData, data: null };

}

const newColor: Color = await response.json();

return { result: true, message: 'Color created', data: newColor };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: "", data: null };

}

}

export async function GetColorById(id: number): Promise<ApiResponse> {

try {

const response = await fetch(`/api/color/GetColorById?id=${id}`, {

method: 'GET',

headers: {

'Content-Type': 'application/json'

},

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData, data: null };

}

const color: Color = await response.json();

return { result: true, message: 'Color fetched', data: color };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: "", data: null };

}

}

export async function UpdateColor(id: number, colorname: string): Promise<ApiResponse> {

try {

const response = await fetch(`/api/color/UpdateColor?id=${id}`, {

method: 'PUT',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify({ colorname })

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData, data: null };

}

const updatedColor: Color = await response.json();

return { result: true, message: 'Color updated', data: updatedColor };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: "", data: null };

}

}

export async function DeleteColor(id: number): Promise<ApiResponse> {

try {

const response = await fetch(`/api/color/DeleteColor?id=${id}`, {

method: 'DELETE',

headers: {

'Content-Type': 'application/json'

},

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData, data: null };

}

return { result: true, message: 'Color deleted', data: null };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: "", data: null };

}

}

export interface Country {

countryid: number;

country: string;

}

interface ApiResponse {

result: boolean,

message?: string;

data?: any;

}

export async function FetchAllCountry(): Promise<ApiResponse> {

try {

const response = await fetch(`/api/country/GetCountries`, {

method: 'GET',

headers: {

'Content-Type': 'application/json'

},

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData, data: [] };

}

const countries: Country[] = await response.json();

return { result: true, message: 'Countries fetched', data: countries};

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: "", data: null };

}

}

export interface SendOrderParams {

email: string;

pdfbase64: string;

Filename: string;

}

interface ApiResponse {

result: boolean;

message?: string;

data?: any;

}

export async function SendOrderEmail({ email, pdfbase64, Filename }: SendOrderParams): Promise<ApiResponse> {

try {

const response = await fetch(`/api/email/SendOrder`, {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({email: email, pdfbase64: pdfbase64, Filename: Filename}),

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData.error, data: null };

}

const message = await response.json();

return { result: true, message: 'Email sent', data: message };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: "", data: null };

}

}

export async function SendEmailToOwner(email:string, text:string, subject:string): Promise<ApiResponse> {

try {

const response = await fetch(`/api/email/SendMessageToOwner`, {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({email: email, text: text, subject: subject}),

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData.error, data: null };

}

const message = await response.json();

return { result: true, message: 'Email sent', data: message };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: "", data: null };

}

}

export async function SendMesssage(email:string, text:string, subject:string): Promise<ApiResponse> {

try {

const response = await fetch(`/api/email/SendMessage`, {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({email: email, text: text, subject: subject}),

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData.error, data: null };

}

const message = await response.json();

return { result: true, message: 'Email sent', data: message };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: "", data: null };

}

}

interface ApiResponse {

result: boolean,

message?: string;

data: string | null;

}

export const generateUniqueImageName = () => {

const timestamp = Date.now();

return `image\_${timestamp}`;

};

export function isValidLink(link: string): boolean {

const urlRegex = /^(https?:\/\/)?([a-zA-Z0-9-]+(\.[a-zA-Z0-9-]+)+.\*)$/;

return urlRegex.test(link);

}

export async function AddCategoryImage(

file: File,

foldername: string,

categoryname: string,

filetype: string

): Promise<ApiResponse> {

try {

const formData = new FormData();

formData.append('file', file);

formData.append('foldername', foldername);

formData.append('categoryname', categoryname);

formData.append('filetype', filetype);

const response = await fetch(`/api/Images/category/POST`, {

method: 'POST',

body: formData,

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error adding category image:', errorData.error);

return { result: false, message: errorData.error, data: null };

}

const newimage = await response.json();

return { result: true, message: 'Category image added successfully', data: newimage };

} catch (error) {

console.error('Failed to add category image:', error);

return { result: false, message: "", data: null };

}

}

export async function DeleteCategoryImage(

categoryname: string,

foldername: string,

): Promise<ApiResponse> {

try {

const response = await fetch(`/api/Images/category/DELETE`, {

method: 'DELETE',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({

categoryname,

foldername

}),

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error deleting category image:', errorData.error);

return { result: false, message: errorData.error, data: null };

}

const newimage = await response.json();

return { result: true, message: 'category image deleted successfully', data: newimage };

} catch (error) {

console.error('Failed to delete category image:', error);

return { result: false, message: "", data: null };

}

}

export async function AddProductImage(

file: File,

categoryname: string,

productid: string,

imagename: string,

filetype: string

): Promise<ApiResponse> {

try {

const formData = new FormData();

formData.append('file', file);

formData.append('categoryname', categoryname);

formData.append('productid', productid);

formData.append('imagename', imagename);

formData.append('filetype', filetype);

const response = await fetch(`/api/Images/product/POST`, {

method: 'POST',

body: formData,

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error adding product image:', errorData.error);

return { result: false, message: errorData.error, data: null };

}

const newimage = await response.json();

return { result: true, message: 'product image added successfully', data: newimage };

} catch (error) {

console.error('Failed to add product image:', error);

return { result: false, message: "", data: null };

}

}

export async function DeleteProductImage(

imageurl: string,

): Promise<ApiResponse> {

try {

const response = await fetch(`/api/Images/product/DELETE`, {

method: 'DELETE',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({

imageurl

}),

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error deleting product image:', errorData.error);

return { result: false, message: errorData.error, data: null };

}

const newimage = await response.json();

return { result: true, message: 'product image deleted successfully', data: newimage };

} catch (error) {

console.error('Failed to delete product image:', error);

return { result: false, message: "", data: null };

}

}

export async function DeleteAllImages(

foldername: string,

): Promise<ApiResponse> {

try {

const response = await fetch(`/api/Images/product/DELETEALL`, {

method: 'DELETE',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({

foldername

}),

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error deleting product images:', errorData.error);

return { result: false, message: errorData.error, data: null };

}

const deletimages = await response.json();

return { result: true, message: 'product image deleted successfully', data: deletimages };

} catch (error) {

console.error('Failed to delete product image:', error);

return { result: false, message: "", data: null };

}

}

export async function AddMediaImage(

file: File,

foldername: string,

medianame: string,

filetype: string

): Promise<ApiResponse> {

try {

const formData = new FormData();

formData.append('file', file);

formData.append('foldername', foldername);

formData.append('medianame', medianame);

formData.append('filetype', filetype);

const response = await fetch(`/api/Images/Media/POST`, {

method: 'POST',

body: formData,

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error adding product image:', errorData.error);

return { result: false, message: errorData.error, data: null };

}

const newimage = await response.json();

return { result: true, message: 'product image added successfully', data: newimage };

} catch (error) {

console.error('Failed to add product image:', error);

return { result: false, message: "", data: null };

}

}

export async function DeleteMediaImage(

medianame: string,

): Promise<ApiResponse> {

try {

const response = await fetch(`/api/Images/Media/DELETE`, {

method: 'DELETE',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({

medianame

}),

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error deleting media image:', errorData.error);

return { result: false, message: errorData.error, data: null };

}

const newimage = await response.json();

return { result: true, message: 'media image deleted successfully', data: newimage };

} catch (error) {

console.error('Failed to delete media image:', error);

return { result: false, message: "", data: null };

}

}

import type { Product } from "./Product";

export interface OrdersProduct {

orderid: number; // ID of the order

productid: number;

productname: string;

colorname: string;

sizename: string;

quantity: number;

købspris\_ex\_moms: number;

salgpris\_ex\_moms: number;

indgående\_moms: number;

udgående\_moms: number;

}

export interface Order {

orderid: number;

order\_date: string; // Using string to represent dates in ISO format

firstname: string;

lastname: string;

email: string;

phonenumber?: number;

address: string;

postalcode: number;

city: string;

country: string;

parcel: string;

ishomedelivery: boolean;

totalprice: number;

stripepaymentid: string;

userid: number;

statusid: number;

statusname?: string;

paymentid: number;

paymentname?: string;

order\_products: OrdersProduct[];

products?: Product[];

}

interface ApiResponse {

result: boolean;

message?: string;

data?: any;

authtoken?: string

}

export async function FetchAllOrders(): Promise<ApiResponse> {

try {

const response = await fetch(`/api/order/GetOrders`, {

method: 'GET',

headers: {

'Content-Type': 'application/json',

},

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData.error, data: [] };

}

const orders: Order[] = await response.json();

return { result: true, message: 'Orders fetched successfully', data: orders };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: "", data: null };

}

}

export async function FetchOrderById(orderid: string): Promise<ApiResponse> {

try {

const response = await fetch(`/api/order/FetchOrderById?id=${orderid}`, {

method: 'GET',

headers: {

'Content-Type': 'application/json',

},

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData.error, data: null };

}

const order: Order = await response.json();

return { result: true, message: 'Order fetched successfully', data: order };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: "", data: null };

}

}

export async function FetchOrdersByUserId(userid: string): Promise<ApiResponse> {

try {

const response = await fetch(`/api/order/FetchOrdersByUserId?id=${userid}`, {

method: 'GET',

headers: {

'Content-Type': 'application/json',

},

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData.error, data: [] };

}

const orders: Order[] = await response.json();

return { result: true, message: 'Orders fetched successfully', data: orders };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: "", data: null };

}

}

export async function CreateOrder(order: Order): Promise<ApiResponse> {

try {

const response = await fetch(`/api/order/CreateOrder`, {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify(order),

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData.error, data: null };

}

const data = await response.json();

const newOrder = data.order;

return { result: true, message: 'Order created successfully', data: newOrder, authtoken: data.jwt };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: "", data: null };

}

}

export async function UpdateOrder(orderid: number, updatedOrder: Omit<Order, 'orderid'>): Promise<ApiResponse> {

try {

const response = await fetch(`/api/order/UpdateOrder?id=${orderid}`, {

method: 'PUT',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify(updatedOrder),

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData.error, data: null };

}

const updated: Order = await response.json();

return { result: true, message: 'Order updated successfully', data: updated };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: "", data: null };

}

}

export async function DeleteOrder(orderid: number): Promise<ApiResponse> {

try {

const response = await fetch(`/api/order/DeleteOrder?id=${orderid}`, {

method: 'DELETE',

headers: {

'Content-Type': 'application/json',

},

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData.error, data: null };

}

return { result: true, message: 'Order deleted successfully', data: null };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: "", data: null };

}

}

export interface ProductImage {

imageid: number;

productid: number;

image\_url: string;

description?: string; // Optional field

created\_at?: string; // Optional field, assuming it's returned as a string in ISO format

}

export interface Category {

categoryid: number;

categoryname: string;

categoryimage: string;

categorydescription: string;

}

export interface ProductColorSize {

colorid: number; // Corresponds to colorid

colorname?: string; // Corresponds to colorname

sizeid: number; // Corresponds to sizeid

sizename?: string; // Corresponds to sizename

quantity: number; // Corresponds to quantity

}

// Update the Product interface

export interface Product {

productid: number;

name: string;

description: string;

manufacturer: string;

features: string;

link: string;

købspris\_ex\_moms: number;

salgpris\_ex\_moms: number;

indgående\_moms: number;

udgående\_moms: number;

tags: string;

barcode: number;

images: ProductImage[]; // Array of ProductImage objects

productcolorsizes: ProductColorSize[];

categories: Category[];

}

interface ApiResponse {

result: boolean,

message?: string;

data?: any;

}

export async function GetAllProducts(): Promise<ApiResponse> {

try {

const response = await fetch(`/api/product/GetAllProducts`, {

method: 'GET',

headers: {

'Content-Type': 'application/json'

},

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData, data: [] };

}

const countries: Product[] = await response.json();

return { result: true, message: 'Products fetched', data: countries};

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: "", data: null };

}

}

export const GetProduct = async (id: string): Promise<ApiResponse> => {

const response = await fetch(`/api/product/GetProduct?id=${id}`);

if (!response.ok) {

return { result: false, message: "Cannot fetch Product", data: null };

}

// Parse the JSON response as a User object

const user: Product = await response.json();

return { result: true, message: "Product fetched", data: user };

};

export const AddProduct = async (product: Product, IsLocalCdn: boolean): Promise<ApiResponse> => {

try {

const response = await fetch(`/api/product/AddProduct?IsLocalCdn=${IsLocalCdn}`, {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify(product),

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData);

return { result: false, message: errorData.message || 'Failed to create product', data: null };

}

const createdProduct: Product = await response.json();

return { result: true, message: 'Product created successfully', data: createdProduct };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: 'Failed to create product', data: null };

}

};

export const DeleteProduct = async (productid: number): Promise<ApiResponse> => {

try {

const response = await fetch(`/api/product/DeleteProduct?id=${productid}`, {

method: 'DELETE',

headers: {

'Content-Type': 'application/json',

},

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData);

return { result: false, message: errorData.message || 'Failed to delete product', data: null };

}

return { result: true, message: 'Product deleted successfully', data: { productid } };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: 'Failed to delete product', data: null };

}

};

export const DeleteProducts = async (startid: number, endid: number): Promise<ApiResponse> => {

try {

const response = await fetch(`/api/product/DeleteProducts?startid=${startid}&endid=${endid}`, {

method: 'DELETE',

headers: {

'Content-Type': 'application/json',

},

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData);

return { result: false, message: errorData.message || 'Failed to delete product', data: null };

}

return { result: true, message: 'Products deleted successfully', data: `${startid}-${endid}` };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: 'Failed to delete product', data: null };

}

};

export const UpdateProduct = async (product: Product): Promise<ApiResponse> => {

try {

const response = await fetch(`/api/product/UpdateProduct?id=${product.productid}`, {

method: 'PUT',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify(product),

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData);

return { result: false, message: errorData.message || 'Failed to update product', data: null };

}

const updatedProduct: Product = await response.json();

return { result: true, message: 'Product updated successfully', data: updatedProduct };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: 'Failed to update product', data: null };

}

};

interface ProductCategory {

productId: number;

categoryId: number;

}

interface ApiResponse {

result: boolean,

message?: string;

data?: any;

}

export async function GetProductCategories(): Promise<ApiResponse> {

try {

const response = await fetch(`/api/productCategory/GetProductCategories`, {

method: 'GET',

headers: {

'Content-Type': 'application/json'

},

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData, data: [] };

}

const countries: ProductCategory[] = await response.json();

return { result: true, message: 'ProductCategory fetched', data: countries};

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: "", data: null };

}

}

export async function GetProductCategoriesByCategoryId(categoryid: number): Promise<ApiResponse> {

try {

const response = await fetch(`/api/productCategory/GetProductCategoriesByCategoryId?id=${categoryid}`, {

method: 'GET',

headers: {

'Content-Type': 'application/json'

},

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData, data: [] };

}

const countries: ProductCategory[] = await response.json();

return { result: true, message: 'ProductCategory fetched', data: countries};

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: "", data: null };

}

}

export interface ProductColorSize {

productId: number; // Corresponds to productid in the table

colorId: number; // Corresponds to colorid in the table

sizeId: number; // Corresponds to sizeid in the table

quantity: number; // Corresponds to quantity in the table

}

interface ApiResponse {

result: boolean,

message?: string;

data?: any;

}

export async function UpdateProductColorSizeQuantity(colorid: number, sizeid: number, quantity: number, productid: number): Promise<ApiResponse> {

try {

const response = await fetch(`/api/productColorSize/UpdateProductColorSizeQuantity?colorid=${colorid}&sizeid=${sizeid}&productid=${productid}`, {

method: 'PUT',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify({quantity: quantity}),

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData, data: [] };

}

const productcolorsize: ProductColorSize[] = await response.json();

return { result: true, message: 'Productcolorsize updated', data: productcolorsize};

} catch (error) {

console.error('Productcolorsize failed:', error);

return { result: false, message: "", data: null };

}

}

export async function GetProductColorSizeById(colorid: number, sizeid: number, productid: number): Promise<ApiResponse> {

try {

const response = await fetch(`/api/productColorSize/GetProductColorSizeById?colorid=${colorid}&sizeid=${sizeid}&productid=${productid}`, {

method: 'GET',

headers: {

'Content-Type': 'application/json',

},

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error fetching Productcolorsize:', errorData.error);

return { result: false, message: errorData.error, data: null };

}

const productcolorsize: ProductColorSize = await response.json();

return { result: true, message: 'Productcolorsize FETCHED', data: productcolorsize};

} catch (error) {

console.error('Failed to fetch Productcolorsize:', error);

return { result: false, message: "", data: null };

}

}

export interface ProductImage {

imageid: number;

productid: number;

image\_url: string;

description?: string;

created\_at: string;

}

interface ApiResponse {

result: boolean,

message?: string;

data?: any;

}

// for api routes

export async function GetAllProductImages(productid: number): Promise<ApiResponse> {

try {

const response = await fetch(`http://localhost:5000/api/product-images/${productid}`, {

method: 'GET',

headers: {

'Content-Type': 'application/json'

},

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData.error, data: [] };

}

const images: ProductImage[] = await response.json();

return { result: true, message: 'Images fetched successfully', data: images };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: error instanceof Error ? error.message : 'An error occurred', data: null };

}

}

// Add a new ProductImage entry

export async function AddProductImages(productImage: { productid: number; image\_url: string; description?: string; }): Promise<ApiResponse> {

try {

const response = await fetch(`/api/productImages/CreateProductImage`, {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(productImage)

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData.error };

}

const newImage: ProductImage = await response.json();

return { result: true, message: 'Image added successfully', data: newImage };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: error instanceof Error ? error.message : 'An error occurred' };

}

}

// Update a specific ProductImage entry

export async function UpdateProductImage(imageid: number, updates: { image\_url?: string; description?: string; }): Promise<ApiResponse> {

try {

const response = await fetch(`/api/productImages/UpdateProductImage?id=${imageid}`, {

method: 'PUT',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(updates)

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData.error };

}

const updatedImage: ProductImage = await response.json();

return { result: true, message: 'Image updated successfully', data: updatedImage };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: error instanceof Error ? error.message : 'An error occurred' };

}

}

// Delete a specific ProductImage entry

export async function DeleteProductImages(imageid: number): Promise<ApiResponse> {

try {

const response = await fetch(`/api/productImages/DeleteProductImage?id=${imageid}`, {

method: 'DELETE',

headers: {

'Content-Type': 'application/json'

},

});

if (!response.ok) {

const errorData = await response.json();

console.error('Error:', errorData.error);

return { result: false, message: errorData.error };

}

const deletedImage: ProductImage = await response.json();

return { result: true, message: 'Image deleted successfully', data: deletedImage };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: error instanceof Error ? error.message : 'An error occurred' };

}

}

export interface User {

userid: number; // SERIAL PRIMARY KEY

username: string;

firstname: string; // VARCHAR(255) NOT NULL

lastname: string; // VARCHAR(255) NOT NULL

password: string; // VARCHAR(255) NOT NULL

email: string; // VARCHAR(255) UNIQUE NOT NULL

phonenumber?: number; // BIGINT (optional)

addressid?: number; // INT UNIQUE (optional)

membertypeid?: number; // INT UNIQUE (optional)

// Additional fields from the JOINs

user\_address?: string; // Address from Address table

address\_postalcode?: number; // Postal code from Address table

address\_city?: string; // City from Address table

country\_name?: string; // Country name from Country table

member\_type?: string; // Member type from MemberType table

}

interface ApiResponse {

result: boolean,

message?: string;

data?: any;

}

export const fetchUsers = async (): Promise<ApiResponse> => {

const response = await fetch('/api/user/fetchUsers');

if (!response.ok) {

return { result: false, message: "Cannot fetch users", data: [] };

}

// Ensure the response is parsed as an array of User objects

const users: User[] = await response.json();

return { result: true, message: "Users fetched", data: users };

};

export const fetchUserById = async (id: number): Promise<ApiResponse> => {

const response = await fetch(`/api/user/fetchUserById?id=${id}`);

if (!response.ok) {

return { result: false, message: "Cannot fetch user", data: null };

}

// Parse the JSON response as a User object

const user: User = await response.json();

return { result: true, message: "User fetched", data: user };

};

export const loginUser = async (email: string, password: string): Promise<Boolean> => {

try {

const response = await fetch(`/api/user/loginUser`, {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({ email, password }),

});

if (!response.ok) {

return false

}

const data = await response.json();

const token = data.token;

// Store the token in localStorage

localStorage.setItem('jwtToken', token);

return true // Return the result and token

} catch (error) {

console.error('Login error:', error);

return false // Return false and null token in case of error

}

};

export const changepassword = async (id: string, kodeord: string, nykode: string): Promise<ApiResponse> => {

try {

const response = await fetch(`/api/user/changepassword?id=${id}`, {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({ kodeord, nykode }),

});

if (!response.ok) {

return { result: false, message: "Cannot change password", data: [] };

}

return { result: true, message: "Password changed", data: await response.json() };

} catch (error) {

console.error('Login error:', error);

return { result: false, message: "Cannot change password", data: [] };

}

};

export const RegisterUser = async (username: string, firstname: string, lastname: string,

password: string, email: string, phonenumber: number, addressid: number, membertypeid: number): Promise<ApiResponse> => {

try {

const response = await fetch('/api/user/CreateUser', {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify({ username, firstname, lastname, password, email, phonenumber, addressid, membertypeid }),

});

if (!response.ok) {

const errorData = await response.json();

return { result: false, message: errorData, data: null };

}

const data = await response.json();

return { result: true, message: "", data: data };

} catch (error) {

console.error('Request failed:', error);

return { result: false, message: "Database error", data: null };

}

};

export const UserExsistByEmail = async (email: string): Promise<ApiResponse> => {

const response = await fetch(`/api/user/fetchUserByEmail?email=${email}`, {

method: 'GET',

headers: {

'Content-Type': 'application/json'

},

});

if (!response.ok) {

const errorData = await response.json();

return { result: false, message: errorData, data: null };

}

const data: User = await response.json();

return { result: true, message: "", data: data };

}

export const IsLoggedIn = async (): Promise<ApiResponse> => {

const token = localStorage.getItem('jwtToken');

if (token) {

const response = await fetch('/api/user/verifyLogin', {

method: 'GET',

headers: {

'Authorization': `Bearer ${token}`

}

});

if (response.ok) {

const data = await response.json();

return { result: true, message: "User is logged in", data: data };

} else {

return { result: false, message: "User is not logged in", data: '' };

}

} else {

return { result: false, message: "User is logged in", data: '' };

}

};

export const UpdateUser = async ( user: User

): Promise<ApiResponse> => {

try {

const response = await fetch(`/api/user/UpdateUser?id=${user.userid}`, {

method: 'PUT',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify(user),

});

if (!response.ok) {

const errorData = await response.json();

return { result: false, message: errorData, data: null };

}

const data = await response.json();

return { result: true, message: "User updated successfully", data };

} catch (error) {

console.error('Update error:', error);

return { result: false, message: "Database error", data: null };

}

};

export const deleteUser = async (id: number): Promise<ApiResponse> => {

try {

const response = await fetch(`/api/user/deleteUser?id=${id}`, {

method: 'DELETE',

});

if (response.status === 204) {

return { result: true, message: "User deleted successfully" };

} else if (response.status === 404) {

return { result: false, message: "User not found" };

} else {

const errorData = await response.json();

return { result: false, message: errorData.error || "An error occurred" };

}

} catch (error) {

console.error("Error deleting user:", error);

return { result: false, message: "Server error" };

}

};

export const RecoverUser = async (id: string): Promise<ApiResponse> => {

try {

const response = await fetch(`/api/user/recoverUser?id=${id}`, {

method: 'POST',

headers: {

'Content-Type': 'application/json',

}

});

if (!response.ok) {

return { result: false, message: "Cannot get new password", data: [] };

}

return { result: true, message: "New password generated", data: await response.json() };

} catch (error) {

console.error('Login error:', error);

return { result: false, message: "Cannot get new password", data: [] };

}

};

# **Api**

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

export const POST: APIRoute = async ({ request }) => {

const { address, postalcode, city, countryid } = await request.json();

try {

if (pool.connection) {

const result = await pool.pool.query(

'INSERT INTO Address (address, postalcode, city, countryid) VALUES ($1, $2, $3, $4) RETURNING \*',

[address, postalcode, city, countryid]

);

return new Response(

JSON.stringify({ addressid: result.rows[0].addressid }),

{

status: 201,

headers: {

'Content-Type': 'application/json',

},

}

);

} else {

console.error('Failed to connect to the database');

return new Response(

JSON.stringify({ error: 'Failed to connect to the database' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

} catch (err) {

console.error('Database error', err);

return new Response(

JSON.stringify({ error: 'Database error' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

};

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

// Update an address by ID

export const PUT: APIRoute = async ({ url, request }) => {

const urlSearchParams = new URLSearchParams(url.search);

const id = urlSearchParams.get('id');

const { address, postalcode, city, countryid } = await request.json();

try {

if (pool.connection) {

const result = await pool.pool.query(

'UPDATE Address SET address = $1, postalcode = $2, city = $3, countryid = $4 WHERE addressid = $5 RETURNING \*',

[address, postalcode, city, countryid, id]

);

if (result.rows.length === 0) {

return new Response(

JSON.stringify({ error: 'Address not found' }),

{

status: 404,

headers: {

'Content-Type': 'application/json',

},

}

);

}

return new Response(

JSON.stringify(result.rows[0]),

{

status: 200,

headers: {

'Content-Type': 'application/json',

},

}

);

} else {

console.error('Failed to connect to the database');

return new Response(

JSON.stringify({ error: 'Failed to connect to the database' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

} catch (err) {

console.error('Database error', err);

return new Response(

JSON.stringify({ error: 'Database error' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

};

// src/pages/api/categories/create.json.ts

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

// Create a new category

export const POST: APIRoute = async ({ request }) => {

const { categoryname, categoryimage, categorydescription } = await request.json();

try {

const result = await pool.pool.query(

'INSERT INTO Category (categoryname, categoryimage, categorydescription) VALUES ($1, $2, $3) RETURNING \*',

[categoryname, categoryimage, categorydescription]

);

return new Response(

JSON.stringify(result.rows[0]),

{

status: 201,

headers: {

'Content-Type': 'application/json',

},

}

);

} catch (error) {

console.error('Error creating category:', error);

return new Response(

JSON.stringify({ error: 'Internal Server Error' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

};

// src/pages/api/categories/[id]/delete.json.ts

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

// Delete a category by ID

export const DELETE: APIRoute = async ({ url }) => {

const urlSearchParams = new URLSearchParams(url.search);

const id = urlSearchParams.get('id');

try {

const result = await pool.pool.query('DELETE FROM Category WHERE categoryid = $1 RETURNING \*', [id]);

if (result.rows.length > 0) {

return new Response(

JSON.stringify({ message: 'Category deleted successfully' }),

{

status: 200,

headers: {

'Content-Type': 'application/json',

},

}

);

} else {

return new Response(

JSON.stringify({ error: 'Category not found' }),

{

status: 404,

headers: {

'Content-Type': 'application/json',

},

}

);

}

} catch (error) {

console.error('Error deleting category:', error);

return new Response(

JSON.stringify({ error: 'Internal Server Error' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

};

// src/pages/api/categories/index.json.ts

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

// Read all categories

export const GET: APIRoute = async () => {

try {

const result = await pool.pool.query('SELECT \* FROM Category');

return new Response(

JSON.stringify(result.rows),

{

status: 200,

headers: {

'Content-Type': 'application/json',

},

}

);

} catch (error) {

console.error('Error reading categories:', error);

return new Response(

JSON.stringify({ error: 'Internal Server Error' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

};

// src/pages/api/categories/[id].json.ts

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

// Read a single category by ID

export const GET: APIRoute = async ({ url }) => {

const urlSearchParams = new URLSearchParams(url.search);

const id = urlSearchParams.get('id');

try {

const result = await pool.pool.query('SELECT \* FROM Category WHERE categoryid = $1', [id]);

if (result.rows.length > 0) {

return new Response(

JSON.stringify(result.rows[0]),

{

status: 200,

headers: {

'Content-Type': 'application/json',

},

}

);

} else {

return new Response(

JSON.stringify({ error: 'Category not found' }),

{

status: 404,

headers: {

'Content-Type': 'application/json',

},

}

);

}

} catch (error) {

console.error('Error reading category:', error);

return new Response(

JSON.stringify({ error: 'Internal Server Error' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

};

// src/pages/api/categories/[id]/update.json.ts

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

// Update a category by ID

export const PUT: APIRoute = async ({ request, url }) => {

const urlSearchParams = new URLSearchParams(url.search);

const id = urlSearchParams.get('id');

const { categoryname, categoryimage, categorydescription } = await request.json();

try {

const result = await pool.pool.query(

'UPDATE Category SET categoryname = $1, categoryimage = $2, categorydescription = $3 WHERE categoryid = $4 RETURNING \*',

[categoryname, categoryimage, categorydescription, id]

);

if (result.rows.length > 0) {

return new Response(

JSON.stringify(result.rows[0]),

{

status: 200,

headers: {

'Content-Type': 'application/json',

},

}

);

} else {

return new Response(

JSON.stringify({ error: 'Category not found' }),

{

status: 404,

headers: {

'Content-Type': 'application/json',

},

}

);

}

} catch (error) {

console.error('Error updating category:', error);

return new Response(

JSON.stringify({ error: 'Internal Server Error' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

};

// src/pages/api/colors/create.json.ts

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

// Create a new color

export const POST: APIRoute = async ({ request }) => {

const { colorname } = await request.json();

try {

const result = await pool.pool.query(

'INSERT INTO Color (colorname) VALUES ($1) RETURNING \*',

[colorname]

);

return new Response(

JSON.stringify(result.rows[0]),

{

status: 201,

headers: {

'Content-Type': 'application/json',

},

}

);

} catch (error) {

console.error('Error creating color:', error);

return new Response(

JSON.stringify({ error: 'Failed to create color' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

};

// src/pages/api/colors/[id]/delete.json.ts

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

// Delete a color by ID

export const DELETE: APIRoute = async ({ url }) => {

const urlSearchParams = new URLSearchParams(url.search);

const id = urlSearchParams.get('id');

try {

const result = await pool.pool.query('DELETE FROM Color WHERE colorid = $1 RETURNING \*', [id]);

if (result.rows.length > 0) {

return new Response(

JSON.stringify({ message: 'Color deleted successfully' }),

{

status: 204,

headers: {

'Content-Type': 'application/json',

},

}

);

} else {

return new Response(

JSON.stringify({ error: 'Color not found' }),

{

status: 404,

headers: {

'Content-Type': 'application/json',

},

}

);

}

} catch (error) {

console.error('Error deleting color:', error);

return new Response(

JSON.stringify({ error: 'Failed to delete color' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

};

// src/pages/api/colors/[id].json.ts

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

// Get a color by ID

export const GET: APIRoute = async ({ url }) => {

const urlSearchParams = new URLSearchParams(url.search);

const id = urlSearchParams.get('id');

try {

const result = await pool.pool.query('SELECT \* FROM Color WHERE colorid = $1', [id]);

if (result.rows.length > 0) {

return new Response(

JSON.stringify(result.rows[0]),

{

status: 200,

headers: {

'Content-Type': 'application/json',

},

}

);

} else {

return new Response(

JSON.stringify({ error: 'Color not found' }),

{

status: 404,

headers: {

'Content-Type': 'application/json',

},

}

);

}

} catch (error) {

console.error('Error retrieving color:', error);

return new Response(

JSON.stringify({ error: 'Failed to retrieve color' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

};

// src/pages/api/colors/index.json.ts

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

// Get all colors

export const GET: APIRoute = async () => {

try {

const result = await pool.pool.query('SELECT \* FROM Color');

return new Response(

JSON.stringify(result.rows),

{

status: 200,

headers: {

'Content-Type': 'application/json',

},

}

);

} catch (error) {

console.error('Error retrieving colors:', error);

return new Response(

JSON.stringify({ error: 'Failed to retrieve colors' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

};

// src/pages/api/colors/[id]/update.json.ts

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

// Update a color by ID

export const PUT: APIRoute = async ({ request, url }) => {

const urlSearchParams = new URLSearchParams(url.search);

const id = urlSearchParams.get('id');

const { colorname } = await request.json();

try {

const result = await pool.pool.query(

'UPDATE Color SET colorname = $1 WHERE colorid = $2 RETURNING \*',

[colorname, id]

);

if (result.rows.length > 0) {

return new Response(

JSON.stringify(result.rows[0]),

{

status: 200,

headers: {

'Content-Type': 'application/json',

},

}

);

} else {

return new Response(

JSON.stringify({ error: 'Color not found' }),

{

status: 404,

headers: {

'Content-Type': 'application/json',

},

}

);

}

} catch (error) {

console.error('Error updating color:', error);

return new Response(

JSON.stringify({ error: 'Failed to update color' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

};

// src/pages/api/Country/index.json.ts

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

// Read all countries

export const GET: APIRoute = async () => {

try {

if (pool.connection) {

const result = await pool.pool.query('SELECT \* FROM Country');

return new Response(

JSON.stringify(result.rows),

{

status: 200,

headers: {

'Content-Type': 'application/json',

},

}

);

} else {

console.error('Failed to connect to the database');

return new Response(

JSON.stringify({ error: 'Failed to connect to the database' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

} catch (err) {

console.error('Database error', err);

return new Response(

JSON.stringify({ error: 'Database error' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

};

import type { APIRoute } from 'astro';

import { v2 as cloudinary } from 'cloudinary';

cloudinary.config({

cloud\_name: import.meta.env.CLOUDINARY\_CLOUD\_NAME,

api\_key: import.meta.env.CLOUDINARY\_API\_KEY,

api\_secret: import.meta.env.CLOUDINARY\_API\_SECRET,

});

export const DELETE: APIRoute = async ({ request }) => {

try {

const { categoryname, foldername }:

{categoryname: string, foldername: string} = await request.json();

if (!categoryname) {

return new Response(

JSON.stringify({ error: 'Public ID is required' }),

{ status: 400, headers: { 'Content-Type': 'application/json' } }

);

}

// Delete the image from Cloudinary

const result = await cloudinary.uploader.destroy(`${foldername}/${categoryname}`, {

invalidate: true, // Optional: Invalidate cached versions of the file

});

if (result.result === 'ok') {

return new Response(

JSON.stringify({

message: 'Image deleted successfully',

}),

{ status: 200, headers: { 'Content-Type': 'application/json' } }

);

} else {

return new Response(

JSON.stringify({ error: 'Failed to delete image' }),

{ status: 500, headers: { 'Content-Type': 'application/json' } }

);

}

} catch (error) {

console.error('Error deleting image:', error);

return new Response(

JSON.stringify({ error: 'Internal Server Error' }),

{ status: 500, headers: { 'Content-Type': 'application/json' } }

);

}

};

import type { APIRoute } from 'astro';

import { v2 as cloudinary } from 'cloudinary';

cloudinary.config({

cloud\_name: import.meta.env.CLOUDINARY\_CLOUD\_NAME,

api\_key: import.meta.env.CLOUDINARY\_API\_KEY,

api\_secret: import.meta.env.CLOUDINARY\_API\_SECRET,

});

export const POST: APIRoute = async ({ request }) => {

try {

const formData = await request.formData();

const file = formData.get('file') as File;

const foldername = formData.get('foldername') as string;

const categoryname = formData.get('categoryname') as string;

const filetype = formData.get('filetype') as string;

if (!file || file.type.split('/')[0] !== 'image') {

return new Response(

JSON.stringify({ error: 'Invalid image file' }),

{ status: 400, headers: { 'Content-Type': 'application/json' } }

);

}

const fileBuffer = await file.arrayBuffer();

const base64Image = `data:${file.type};base64,${Buffer.from(fileBuffer).toString('base64')}`;

// Upload to Cloudinary with folder and replacement enabled

const result = await cloudinary.uploader.upload(base64Image, {

folder: foldername, // Specify the folder

format: filetype, // Convert to .jpg

overwrite: true, // Replace if it already exists

public\_id: categoryname, // Use file name as the public\_id

});

return new Response(

JSON.stringify(result.secure\_url),

{ status: 200, headers: { 'Content-Type': 'application/json' } }

);

} catch (error) {

console.error('Error uploading image:', error);

return new Response(

JSON.stringify({ error: 'Internal Server Error' }),

{ status: 500, headers: { 'Content-Type': 'application/json' } }

);

}

};

import type { APIRoute } from 'astro';

import { v2 as cloudinary } from 'cloudinary';

cloudinary.config({

cloud\_name: import.meta.env.CLOUDINARY\_CLOUD\_NAME,

api\_key: import.meta.env.CLOUDINARY\_API\_KEY,

api\_secret: import.meta.env.CLOUDINARY\_API\_SECRET,

});

export const DELETE: APIRoute = async ({ request }) => {

try {

const { imageurl } = await request.json();

if (!imageurl) {

return new Response(

JSON.stringify({ error: 'Image URL is required' }),

{ status: 400, headers: { 'Content-Type': 'application/json' } }

);

}

// Extract public\_id from image URL

const regex = /\/image\/upload\/(?:v\d+\/)?(.+?)\.(jpg|png|jpeg|gif|webp|bmp)/;

const match = imageurl.match(regex);

if (!match) {

return new Response(

JSON.stringify({ error: 'Invalid Cloudinary image URL' }),

{ status: 400, headers: { 'Content-Type': 'application/json' } }

);

}

const publicId = match[1]; // Extracted public\_id

console.log(publicId)

// Delete the image from Cloudinary

const result = await cloudinary.uploader.destroy(publicId, {

invalidate: true, // Optional: Invalidate cached versions of the file

});

if (result.result === 'ok') {

return new Response(

JSON.stringify({

message: 'Image deleted successfully',

}),

{ status: 200, headers: { 'Content-Type': 'application/json' } }

);

} else {

return new Response(

JSON.stringify({ error: 'Failed to delete image' }),

{ status: 500, headers: { 'Content-Type': 'application/json' } }

);

}

} catch (error) {

console.error('Error deleting image:', error);

return new Response(

JSON.stringify({ error: 'Internal Server Error' }),

{ status: 500, headers: { 'Content-Type': 'application/json' } }

);

}

};

import type { APIRoute } from 'astro';

import { v2 as cloudinary } from 'cloudinary';

cloudinary.config({

cloud\_name: import.meta.env.CLOUDINARY\_CLOUD\_NAME,

api\_key: import.meta.env.CLOUDINARY\_API\_KEY,

api\_secret: import.meta.env.CLOUDINARY\_API\_SECRET,

});

export const DELETE: APIRoute = async ({ request }) => {

try {

const { foldername } = await request.json();

if (!foldername) {

return new Response(

JSON.stringify({ error: 'Image folder is required' }),

{ status: 400, headers: { 'Content-Type': 'application/json' } }

);

}

// Wrap the callback-based API in a promise to allow async handling

const deleteImages = () => {

return new Promise((resolve, reject) => {

cloudinary.api.delete\_resources\_by\_prefix(`${foldername}/`, (error, result) => {

if (error) {

reject(error);

} else {

resolve(result);

}

});

});

};

// Await the deleteImages function to complete before responding

await deleteImages();

return new Response(

JSON.stringify({

message: 'Images deleted successfully',

}),

{ status: 200, headers: { 'Content-Type': 'application/json' } }

);

} catch (error) {

console.error('Error deleting image:', error);

return new Response(

JSON.stringify({ error: 'Internal Server Error' }),

{ status: 500, headers: { 'Content-Type': 'application/json' } }

);

}

};

import type { APIRoute } from 'astro';

import { v2 as cloudinary } from 'cloudinary';

cloudinary.config({

cloud\_name: import.meta.env.CLOUDINARY\_CLOUD\_NAME,

api\_key: import.meta.env.CLOUDINARY\_API\_KEY,

api\_secret: import.meta.env.CLOUDINARY\_API\_SECRET,

});

export const POST: APIRoute = async ({ request }) => {

try {

const formData = await request.formData();

const file = formData.get('file') as File;

const categoryname = formData.get('categoryname') as string;

const productid = formData.get('productid') as string;

const imagename = formData.get('imagename') as string;

const filetype = formData.get('filetype') as string;

if (!file || file.type.split('/')[0] !== 'image') {

return new Response(

JSON.stringify({ error: 'Invalid image file' }),

{ status: 400, headers: { 'Content-Type': 'application/json' } }

);

}

const fileBuffer = await file.arrayBuffer();

const base64Image = `data:${file.type};base64,${Buffer.from(fileBuffer).toString('base64')}`;

// Upload to Cloudinary with folder and replacement enabled

const result = await cloudinary.uploader.upload(base64Image, {

folder: `${categoryname}/${productid}`, // Specify the folder

format: filetype, // Convert to .jpg

overwrite: true, // Replace if it already exists

public\_id: imagename, // Use file name as the public\_id

});

return new Response(

JSON.stringify(result.secure\_url),

{ status: 200, headers: { 'Content-Type': 'application/json' } }

);

} catch (error) {

console.error('Error uploading image:', error);

return new Response(

JSON.stringify({ error: 'Internal Server Error' }),

{ status: 500, headers: { 'Content-Type': 'application/json' } }

);

}

};

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

import jwt from 'jsonwebtoken';

export const POST: APIRoute = async ({ request }) => {

const {

firstname,

lastname,

email,

phonenumber,

address,

postalcode,

city,

country,

parcel,

ishomedelivery,

totalprice,

stripepaymentid,

userid,

statusid,

paymentid,

order\_products

} = await request.json();

if (pool.connection) {

const client = await pool.pool.connect();

try {

await client.query('BEGIN');

const orderDate = new Date().toISOString().split('T')[0];

const orderResult = await client.query(

`INSERT INTO Orders (order\_date, firstname, lastname, email, phonenumber, address, postalcode, city, country, parcel, ishomedelivery, totalprice, stripepaymentid, userid, statusid, paymentid)

VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, $12, $13, $14, $15, $16) RETURNING \*`,

[orderDate, firstname, lastname, email, phonenumber, address, postalcode, city, country, parcel, ishomedelivery, totalprice, stripepaymentid, userid, statusid, paymentid]

);

const newOrderId = orderResult.rows[0].orderid;

if (Array.isArray(order\_products) && order\_products.length > 0) {

const ordersProductInsertPromises = order\_products.map((product) => {

if (!product.productid || !product.colorname || !product.sizename || !product.quantity) {

throw new Error('Invalid OrdersProduct data: missing required fields');

}

return client.query(

`INSERT INTO Orders\_Product (orderid, productid, productname, colorname, sizename, quantity, købspris\_ex\_moms, salgpris\_ex\_moms, indgående\_moms, udgående\_moms)

VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10)`,

[newOrderId, product.productid, product.productname, product.colorname, product.sizename, product.quantity, product.købspris\_ex\_moms, product.salgpris\_ex\_moms, product.indgående\_moms, product.udgående\_moms]

);

});

await Promise.all(ordersProductInsertPromises);

}

await client.query('COMMIT');

const token = jwt.sign(

{ order: orderResult.rows[0] },

import.meta.env.JWTSECRETKEY,

{ expiresIn: "1m" }

);

return new Response(

JSON.stringify({ order: orderResult.rows[0], jwt: token }),

{

status: 201,

headers: {

'Content-Type': 'application/json'

}

}

);

} catch (error) {

await client.query('ROLLBACK');

console.error('Error creating order:', error);

return new Response(

JSON.stringify({ error: 'Failed to create order' }),

{

status: 500,

headers: {

'Content-Type': 'application/json'

}

}

);

} finally {

client.release();

}

} else {

return new Response(

JSON.stringify({ error: 'Database connection failed' }),

{

status: 500,

headers: {

'Content-Type': 'application/json'

}

}

);

}

};

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

export const DELETE: APIRoute = async ({ url }) => {

const urlSearchParams = new URLSearchParams(url.search);

const id = urlSearchParams.get('id');

try {

const result = await pool.pool.query(

'DELETE FROM Orders WHERE orderid = $1 RETURNING \*',

[id]

);

if (result.rows.length === 0) {

return new Response(

JSON.stringify({ error: 'Order not found' }),

{

status: 404,

headers: {

'Content-Type': 'application/json'

}

}

);

}

return new Response(null, { status: 204 });

} catch (error) {

return new Response(

JSON.stringify({ error: 'Failed to delete order' }),

{

status: 500,

headers: {

'Content-Type': 'application/json'

}

}

);

}

};

// src/pages/api/products/create.json.ts

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

// Create a new product

export const POST: APIRoute = async ({ request, url }) => {

const urlSearchParams = new URLSearchParams(url.search);

const IsLocalCdn = urlSearchParams.get('IsLocalCdn') === 'true';

const {

name,

description,

features,

link,

købspris\_ex\_moms,

salgpris\_ex\_moms,

indgående\_moms,

udgående\_moms,

tags,

barcode,

images, // Array of image objects

productcolorsizes, // Array of color/size/quantity objects

manufacturer, // Array of manufacturer IDs

categories // Array of category IDs

} = await request.json();

if (pool.connection) {

const client = await pool.pool.connect();

try {

await client.query('BEGIN'); // Start a transaction

const productResult = await client.query(

`INSERT INTO Product

(name, description, manufacturer, features, link, købspris\_ex\_moms, salgpris\_ex\_moms, indgående\_moms, udgående\_moms, tags, barcode)

VALUES

($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11)

RETURNING \*`,

[

name,

description,

manufacturer,

features,

link,

købspris\_ex\_moms,

salgpris\_ex\_moms,

indgående\_moms,

udgående\_moms,

tags,

barcode

]

);

const newProductId = productResult.rows[0].productid;

// Insert images

if (images && images.length > 0) {

const imageInsertPromises = images.map((image: { image\_url: string; description: any; }) => {

const newimageurl = `https://res.cloudinary.com/${import.meta.env.CLOUDINARY\_CLOUD\_NAME}/image/upload/v${new Date().getTime()}/${categories[0].categoryname}/${newProductId}/${image.image\_url}.jpg`

return client.query(

`INSERT INTO ProductImages (productid, image\_url, description)

VALUES ($1, $2, $3)`,

[newProductId, IsLocalCdn ? newimageurl : image.image\_url, image.description]

);

});

await Promise.all(imageInsertPromises);

}

// Insert product color/size relationships

if (productcolorsizes && productcolorsizes.length > 0) {

const colorSizeInsertPromises = productcolorsizes.map(({ colorid, sizeid, quantity }: {colorid: number, sizeid: number, quantity: number}) => {

return client.query(

`INSERT INTO ProductColorSize (productid, colorid, sizeid, quantity)

VALUES ($1, $2, $3, $4)`,

[newProductId, colorid, sizeid, quantity]

);

});

await Promise.all(colorSizeInsertPromises);

}

// Insert product-category relationships

if (categories && categories.length > 0) {

const categoryInsertPromises = categories.map(({ categoryid }: {categoryid: number}) => {

return client.query(

`INSERT INTO ProductCategory (productid, categoryid)

VALUES ($1, $2)`,

[newProductId, categoryid]

);

});

await Promise.all(categoryInsertPromises);

}

await client.query('COMMIT'); // Commit the transaction

return new Response(

JSON.stringify({ product: productResult.rows[0] }),

{

status: 201,

headers: {

'Content-Type': 'application/json',

},

}

);

} catch (error) {

await client.query('ROLLBACK');

console.error('Error creating product with relations:', error);

return new Response(

JSON.stringify({ error: 'Failed to create product with relations' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

} finally {

client.release();

}

} else {

console.error('Failed to connect to the database');

return new Response(

JSON.stringify({ error: 'Failed to connect to the database' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

};

// src/pages/api/products/[id]/delete.json.ts

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

// Delete a product by ID

export const DELETE: APIRoute = async ({ url }) => {

const urlSearchParams = new URLSearchParams(url.search);

const id = urlSearchParams.get('id');

try {

if (pool.connection) {

const query = `DELETE FROM Product WHERE productid = $1 RETURNING \*`;

const result = await pool.pool.query(query, [id]);

if (result.rowCount === 0) {

console.error('Product not found');

return new Response(

JSON.stringify({ error: 'Product not found' }),

{

status: 404,

headers: {

'Content-Type': 'application/json',

},

}

);

}

return new Response(

JSON.stringify({ message: 'Product deleted successfully' }),

{

status: 200,

headers: {

'Content-Type': 'application/json',

},

}

);

} else {

console.error('Failed to connect to the database');

return new Response(

JSON.stringify({ error: 'Failed to connect to the database' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

} catch (error) {

console.error('Database error', error);

return new Response(

JSON.stringify({ error: 'Database error' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

};

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

// Read all product categories

export const GET: APIRoute = async () => {

try {

if (pool.connection) {

// Query to retrieve all ProductCategory entries

const result = await pool.pool.query('SELECT \* FROM ProductCategory');

return new Response(

JSON.stringify(result.rows),

{

status: 200,

headers: {

'Content-Type': 'application/json',

},

}

);

} else {

console.error('Failed to connect to the database');

return new Response(

JSON.stringify({ error: 'Failed to connect to the database' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

} catch (err) {

console.error('Database error', err);

return new Response(

JSON.stringify({ error: 'Database error' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

};

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

// Read product categories by category ID

export const GET: APIRoute = async ({ url }) => {

const urlSearchParams = new URLSearchParams(url.search);

const categoryid = urlSearchParams.get('id');

try {

// Validate categoryid

if (!categoryid) {

return new Response(

JSON.stringify({ error: 'categoryid is required' }),

{

status: 400,

headers: {

'Content-Type': 'application/json',

},

}

);

}

// Query to fetch product categories by categoryid

const result = await pool.pool.query(

'SELECT \* FROM ProductCategory WHERE categoryid = $1',

[categoryid]

);

if (result.rows.length > 0) {

return new Response(

JSON.stringify(result.rows),

{

status: 200,

headers: {

'Content-Type': 'application/json',

},

}

);

} else {

return new Response(

JSON.stringify({ error: 'No product categories found for the given categoryid' }),

{

status: 404,

headers: {

'Content-Type': 'application/json',

},

}

);

}

} catch (error) {

console.error('Error reading product categories:', error);

return new Response(

JSON.stringify({ error: 'Internal Server Error' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

};

// src/pages/api/categories/[id].json.ts

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

// Read a single category by ID

export const GET: APIRoute = async ({ url }) => {

const urlSearchParams = new URLSearchParams(url.search);

const colorid = urlSearchParams.get('colorid');

const sizeid = urlSearchParams.get('sizeid');

const productid = urlSearchParams.get('productid');

try {

const result = await pool.pool.query('SELECT \* FROM ProductColorSize WHERE colorid = $1 AND sizeid = $2 AND productid = $3', [colorid, sizeid, productid]);

if (result.rows.length > 0) {

return new Response(

JSON.stringify(result.rows[0]),

{

status: 200,

headers: {

'Content-Type': 'application/json',

},

}

);

} else {

return new Response(

JSON.stringify({ error: 'ProductColorSize not found' }),

{

status: 404,

headers: {

'Content-Type': 'application/json',

},

}

);

}

} catch (error) {

console.error('Error reading category:', error);

return new Response(

JSON.stringify({ error: 'Internal Server Error' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

};

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

export const PUT: APIRoute = async ({ request, url }) => {

const urlSearchParams = new URLSearchParams(url.search);

const colorid = urlSearchParams.get('colorid');

const sizeid = urlSearchParams.get('sizeid');

const productid = urlSearchParams.get('productid');

const {

quantity

} = await request.json();

try {

const result = await pool.pool.query(

`UPDATE ProductColorSize

SET quantity = $1

WHERE colorid = $2 AND sizeid = $3 AND productid = $4

RETURNING \*`,

[quantity, colorid, sizeid, productid]

);

if (result.rows.length === 0) {

return new Response(

JSON.stringify({ error: 'productcolorsize not found' }),

{

status: 404,

headers: {

'Content-Type': 'application/json'

}

}

);

}

return new Response(

JSON.stringify(result.rows[0]),

{

status: 200,

headers: {

'Content-Type': 'application/json'

}

}

);

} catch (error) {

return new Response(

JSON.stringify({ error: 'Failed to update productcolorsize quantity' }),

{

status: 500,

headers: {

'Content-Type': 'application/json'

}

}

);

}

};

// src/pages/api/product-images/index.json.ts

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

// Create a new ProductImage entry

export const POST: APIRoute = async ({ request }) => {

const { productid, image\_url, description } = await request.json();

try {

const query = `

INSERT INTO ProductImages (productid, image\_url, description)

VALUES ($1, $2, $3)

RETURNING \*;

`;

const result = await pool.pool.query(query, [productid, image\_url, description]);

return new Response(

JSON.stringify(result.rows[0]),

{

status: 201,

headers: { 'Content-Type': 'application/json' }

}

);

} catch (err) {

console.error('Database error:', err);

return new Response(

JSON.stringify({ error: 'Database error' }),

{

status: 500,

headers: { 'Content-Type': 'application/json' }

}

);

}

};

// src/pages/api/product-images/[imageid]/delete.json.ts

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

// Delete a specific ProductImage entry

export const DELETE: APIRoute = async ({ url }) => {

const urlSearchParams = new URLSearchParams(url.search);

const id = urlSearchParams.get('id');

try {

const query = `

DELETE FROM ProductImages

WHERE imageid = $1

RETURNING \*;

`;

const result = await pool.pool.query(query, [id]);

if (result.rowCount === 0) {

return new Response(

JSON.stringify({ error: 'Image not found' }),

{

status: 404,

headers: { 'Content-Type': 'application/json' }

}

);

}

return new Response(

JSON.stringify(result.rows[0]),

{

status: 200,

headers: { 'Content-Type': 'application/json' }

}

);

} catch (err) {

console.error('Database error:', err);

return new Response(

JSON.stringify({ error: 'Database error' }),

{

status: 500,

headers: { 'Content-Type': 'application/json' }

}

);

}

};

// src/pages/api/sizes/create.json.ts

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

// Create a new size

export const POST: APIRoute = async ({ request }) => {

const { sizename } = await request.json();

try {

const result = await pool.pool.query(

'INSERT INTO Sizes (sizename) VALUES ($1) RETURNING \*',

[sizename]

);

return new Response(

JSON.stringify(result.rows[0]),

{

status: 201,

headers: {

'Content-Type': 'application/json',

},

}

);

} catch (error) {

console.error('Error creating size:', error);

return new Response(

JSON.stringify({ error: 'Failed to create size' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

};

// src/pages/api/sizes/[id]/delete.json.ts

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

// Delete a size by ID

export const DELETE: APIRoute = async ({ url }) => {

const urlSearchParams = new URLSearchParams(url.search);

const id = urlSearchParams.get('id');

try {

const result = await pool.pool.query('DELETE FROM Sizes WHERE sizeid = $1 RETURNING \*', [id]);

if (result.rows.length > 0) {

return new Response(

JSON.stringify({ message: 'Size deleted successfully' }),

{

status: 204,

headers: {

'Content-Type': 'application/json',

},

}

);

} else {

return new Response(

JSON.stringify({ error: 'Size not found' }),

{

status: 404,

headers: {

'Content-Type': 'application/json',

},

}

);

}

} catch (error) {

console.error('Error deleting size:', error);

return new Response(

JSON.stringify({ error: 'Failed to delete size' }),

{

status: 500,

headers: {

'Content-Type': 'application/json',

},

}

);

}

};

// src/pages/api/create-payment-intent.json.ts

import type { APIRoute } from 'astro';

import type {stripe} from '../../../../FrontendRequests/Requests-Api/stripe';

import Stripe from 'stripe';

import dotenv from 'dotenv';

dotenv.config();

// Replace 'your-stripe-secret-key' with your actual Stripe secret key

const stripelib = new Stripe(process.env.PRIVATEKEY as string);

export const POST: APIRoute = async ({ request }) => {

try {

// Parse request body

const requestData: stripe = await request.json();

// Validate input

if (!requestData.amount || !requestData.currency) {

return new Response(

JSON.stringify({ error: 'Amount and currency are required.' }),

{ status: 400, headers: { 'Content-Type': 'application/json' } }

);

}

// Create a payment intent

const paymentIntent = await stripelib.paymentIntents.create({

amount : requestData.amount,

currency : requestData.currency,

automatic\_payment\_methods: { enabled: true },

});

// Return the client secret

return new Response(

JSON.stringify({ clientSecret: paymentIntent.client\_secret }),

{ status: 200, headers: { 'Content-Type': 'application/json' } }

);

} catch (error) {

console.error('Error creating payment intent:', error);

return new Response(

JSON.stringify({ error: error instanceof Error ? error.message : 'Internal Server Error' }),

{ status: 500, headers: { 'Content-Type': 'application/json' } }

);

}

};

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

import bcrypt from 'bcrypt';

export const POST: APIRoute = async ({ request }) => {

const { username, firstname, lastname, password, email, phonenumber, addressid, membertypeid } = await request.json();

try {

if (pool.connection) {

const hashedPassword = await bcrypt.hash(password, 10);

const result = await pool.pool.query(

'INSERT INTO Users (username, firstname, lastname, password, email, phonenumber, addressid, membertypeid) VALUES ($1, $2, $3, $4, $5, $6, $7, $8) RETURNING \*',

[username, firstname, lastname, hashedPassword, email, phonenumber, addressid, membertypeid]

);

return new Response(

JSON.stringify(result.rows[0]),

{

status: 201,

headers: {

"Content-Type": "application/json",

},

}

);

} else {

console.error('Failed to connect to the database');

return new Response(

JSON.stringify({ error: 'Failed to connect to the database' }),

{

status: 500,

headers: {

"Content-Type": "application/json",

},

}

);

}

} catch (err) {

console.error('Database error', err);

return new Response(

JSON.stringify({ error: 'Database error' }),

{

status: 500,

headers: {

"Content-Type": "application/json",

},

}

);

}

};

import type { APIRoute } from 'astro';

import pool from '../../../../Backend/database/dbconnection';

export const DELETE: APIRoute = async ({ url }) => {

const urlSearchParams = new URLSearchParams(url.search);

const id = urlSearchParams.get('id');

try {

if (pool.connection) {

const result = await pool.pool.query('DELETE FROM Users WHERE userid = $1 RETURNING \*', [id]);

if (result.rows.length === 0) {

console.error(`User with id ${id} not found`);

return new Response(

JSON.stringify({ error: 'User not found' }),

{

status: 404,

headers: {

"Content-Type": "application/json",

},

}

);

}

return new Response(

null,

{ status: 204 } // No content to return

);

} else {

console.error('Failed to connect to the database');

return new Response(

JSON.stringify({ error: 'Failed to connect to the database' }),

{

status: 500,

headers: {

"Content-Type": "application/json",

},

}

);

}

} catch (err) {

console.error('Database error', err);

return new Response(

JSON.stringify({ error: 'Database error' }),

{

status: 500,

headers: {

"Content-Type": "application/json",

},

}

);

}

};