import { motion } from 'framer-motion';
import Modal from 'react-modal';
import Button from '@/components/common/Button';
import { WalletsProviders } from '@/services/wallets';
import { Wallet } from '@/types/wallet';
import { generateError, throwNotification } from '@/utils/notification';
import Image from 'next/image';
import { useMemo, useState } from 'react';
import { useDispatch } from 'react-redux';
import PopupModal from '@/components/common/PopupModal';

interface Props {
openKeystoreMenuModal: (bool: boolean) => void;
openLedgerModal: (bool: boolean) => void;
closeModal: () => void;
}

const ConnectModal = ({ openKeystoreMenuModal, openLedgerModal, closeModal }: Props) => {
const dispatch = useDispatch();
const [isChecked, setIsChecked] = useState(false);
const [loading, setIsLoading] = useState(false);
const [selectedWallet, setSelectedWallet] = useState<Wallet | undefined>();

const canConnect = isChecked && selectedWallet;

const handleCheckboxChange = (event: React.ChangeEvent) =>
setIsChecked(event.target.checked);

const connect = async () => {
if (!canConnect || !selectedWallet) return
setIsLoading(true)

PROF

```
try {
  if (selectedWallet.name.toLowerCase() === 'keystore') {
    openKeystoreMenuModal(true)
  } else if (selectedWallet.name.toLowerCase() === 'ledger') {
    openLedgerModal(true)
  } else await selectedWallet.connect(dispatch)
  closeModal()
} catch (e) {
  console.error(e)
  throwNotification(
    generateError('Error occurred trying to connect your wallet')
  )
  closeModal()
}

return Promise.resolve()
```

```
}

const animationVariants = {
hidden: { opacity: 0, y: -100 },
visible: { opacity: 1, y: 0 },
};

const modalStyles = {
content: {
background: 'rgba(255, 255, 255, 0.7)', // Semi-transparent background
border: '1px solid #ccc',
borderRadius: '10px',
width: '60%', // Adjusted width
maxWidth: '400px', // Maximum width, adjust as needed
height: 'fit-content', // Adjusted to content
maxHeight: '90vh', // Optional: Maximum height
margin: 'auto', // Centers the modal horizontally and vertically
display: 'flex', // Flex layout to center content
flexDirection: 'column', // Column layout for internal content
justifyContent: 'space-between',
},
overlay: {
backgroundColor: 'rgba(0, 0, 0, 0.5)', // Dark overlay
backdropFilter: 'blur(5px)', // Blur effect for the background
display: 'flex',
alignItems: 'center', // Align modal vertically in the center
justifyContent: 'center', // Align modal horizontally in the center
},
};
return (
```

PROF

```
<motion.div
initial="hidden"
animate="visible"
variants={animationVariants}
transition={{ duration: 0.5 }}
>
```

Connect Wallet

```
{WalletsProviders().map((wallet: Wallet, index) => {
return (
<div
```

```
className={flex flex-col justify-center items-center gap-4 cursor-pointer h-28
bg-transparent border border-borderUnselected hover:border-borderSelected
hover:scale-105 transition-all duration-300 ease-in-out rounded-20 font-
medium text-sm ${ selectedWallet?.name === wallet.name ? 'border-
borderSelected scale-105 transition-all duration-300' : selectedWallet &&
selectedWallet.name !== wallet.name ? 'opacity-25 transition-all duration-
300' : '' }}
key={index}
onClick={() => setSelectedWallet(wallet)}
>
<Image
className="cursor-pointer"
src={${wallet.icon}}
alt="close-icon"
width="32"
height="32"
/>

{wallet.name}



)
})}


<Button
text={'Connect'}
onClick={connect}
/>

</motion.div>

)
}

export default ConnectModal

import { FC } from 'react'
import styled, { css } from 'styled-components';

const StyledButton = styled.div<{ isActive: boolean; isLoading?: boolean }>`
display: flex;
align-items: center;
justify-content: center;
width: 100%;
height: 3.5rem; // Equivalent to 'h-14'
```

PROF

```
font-weight: bold;
background-color: #B67652; // Adjust the color to fit your theme
color: black;
border-radius: 20px; // Equivalent to 'rounded-20'
margin-top: 0.5rem; // Equivalent to 'mt-2'
transition: all 300ms;
cursor: ${(props) => (props.isActive ? 'pointer' : 'not-allowed')};
opacity: ${(props) => (props.isActive ? 1 : 0.25)};

${(props) =>
props.isLoading &&
css /* Additional styles for loading state */ }

&:hover {
/* Hover styles if applicable */
}
`;

interface Props {
text: string
onClick?: () => void
}

const Button: FC = ({ text, onClick }) => {
return (

{(
```

PROF

)}
{text}

);
};

export default Button

```
        {/* {max && (
          <span
            className="text-pink font-bold cursor-pointer"
            onClick={max}
          >
```

```
            Max
          </span>
        )} */}
```