Spécification Technique pour la page `login.tsx`

1. Introduction

La page `login.tsx` gère l'authentification des utilisateurs. Elle permet à un utilisateur de se connecter en saisissant un nom d'utilisateur et un mot de passe, et navigue vers la page de la boutique après une connexion réussie.

2. Objectifs

- Fournir un formulaire de connexion pour les utilisateurs.
- Valider les informations d'identification des utilisateurs via une API.
- Gérer l'état de la connexion WebSocket.
- Naviguer vers la page de la boutique après une connexion réussie.

3. Dépendances

- `react`
- `react-router-dom`
- `../../stores/authentification`
- `../../context/socketContext`
- `./login.css`

4. Propriétés

Aucune propriété directe n'est passée au composant `Login`.

5. États

- `username` : Le nom d'utilisateur saisi par l'utilisateur.
- `password` : Le mot de passe saisi par l'utilisateur.
- `error` : Un message d'erreur affiché en cas d'échec de la connexion.

6. Méthodes

- `handleLogin`: Gère la soumission du formulaire de connexion. Valide les informations d'identification via une API et navigue vers la page de la boutique si la connexion réussit et que la connexion WebSocket est établie.

7. Hooks

- `useState`: Utilisé pour gérer les états `username`, `password`, et `error`.
- `useNavigate`: Utilisé pour la navigation entre les pages.
- `useSocket`: Utilisé pour accéder à la connexion WebSocket et à son état de connexion.
- `useEffect`: Utilisé pour gérer les effets secondaires comme la navigation vers la page de la boutique lorsque la connexion WebSocket est établie après une tentative de connexion échouée.

8. Fonctionnalités

8.1. Gestion de la Soumission du Formulaire

Lorsque l'utilisateur soumet le formulaire, les informations d'identification sont validées via l'API de connexion. Si la connexion réussit et que la connexion WebSocket est établie, l'utilisateur est redirigé vers la page de la boutique. Sinon, un message d'erreur est affiché.

```
const handleLogin = async (e: React.FormEvent<HTMLFormElement>) => {
    e.preventDefault();

    const result = await login(username, password);
    if (result.status === "ok") {
        if (isConnected) {
            navigate("/shop");
        } else {
            setError("Socket connection failed. Please try again.");
        }
    } else {
        setError("Erreur lors du login");
    }
};
```

8.2. Gestion de la Connexion WebSocket

Un effet est utilisé pour surveiller l'état de la connexion WebSocket. Si la connexion est établie après une tentative de connexion échouée, l'utilisateur est redirigé vers la page de la boutique et le message d'erreur est réinitialisé.

```
useEffect(() => {
    if (isConnected && error) {
        setError('');
        navigate("/shop");
    }
}, [isConnected, error, navigate]);
```

9. Structure du Composant

```
import React, { useState, useEffect } from "react";
import { useNavigate } from "react-router-dom";
import { login } from "../../stores/authentification";
import { useSocket } from "../../context/socketContext";
import "./login.css";

export const Login = () => {
    const [username, setUsername] = useState('');
    const [password, setPassword] = useState('');
    const [error, setError] = useState('');
    const { socket, isConnected } = useSocket();
    const navigate = useNavigate();

    const handleLogin = async (e: React.FormEvent<HTMLFormElement>)
=> {
        e.preventDefault();

        const result = await login(username, password);
        if (result.status === "ok") {
            if (isConnected) {
                navigate("/shop");
            } else {
                setError("Socket connection failed. Please try
again.");
            }
        } else {
            setError("Erreur lors du login");
        }
    };

    useEffect(() => {
        if (isConnected && error) {
            setError('');
            navigate("/shop");
        }
    }, [isConnected, error, navigate]);

    return (
        <div className="login-container">
```

```
            <h1 className="login-header">Connect To Pirate
Emporium</h1>
            <form onSubmit={handleLogin}>
                <div className="login-input">
                    <label htmlFor="username">Username :</label>
                    <input
                        type="text"
                        id="username"
                        value={username}
                        onChange={(e) =>
setUsername(e.target.value)}
                        required
                    />
                </div>
                <div className="login-input">
                    <label htmlFor="password">Password :</label>
                    <input
                        type="password"
                        id="password"
                        value={password}
                        onChange={(e) =>
setPassword(e.target.value)}
                        required
                    />
                </div>
                {error && <p className="error-message">{error}</p>}
                <button type="submit"
className="login-button">Connect</button>
            </form>
        </div>
    );
};
```

10. Conclusion

La page `login.tsx` fournit une interface utilisateur simple et efficace pour l'authentification des utilisateurs. Elle gère les interactions avec l'API de connexion et la connexion WebSocket, et assure une navigation fluide vers la page de la boutique après une connexion réussie.