**Complete React Frontend Setup - Store Rating System**

**1. Initial Setup and Dependencies**

**Frontend Project Initialization**

# Navigate to project root and create frontend

cd store-rating-system

npx create-react-app frontend

cd frontend

# Install required dependencies

npm install axios react-router-dom

npm install lucide-react react-hook-form @hookform/resolvers yup

npm install tailwindcss postcss autoprefixer @tailwindcss/forms

npm install react-toastify

# Initialize Tailwind CSS

npx tailwindcss init -p

**Tailwind Configuration**

Update tailwind.config.js:

/\*\* @type {import('tailwindcss').Config} \*/

module.exports = {

content: ["./src/\*\*/\*.{js,jsx,ts,tsx}"],

theme: {

extend: {

colors: {

primary: {

50: '#eff6ff',

500: '#3b82f6',

600: '#2563eb',

700: '#1d4ed8',

},

},

},

},

plugins: [require('@tailwindcss/forms')],

}

Update src/index.css:

@tailwind base;

@tailwind components;

@tailwind utilities;

@layer components {

.btn-primary {

@apply bg-primary-600 hover:bg-primary-700 text-white font-medium py-2 px-4 rounded-lg transition-colors focus:outline-none focus:ring-2 focus:ring-primary-500 focus:ring-offset-2;

}

.btn-secondary {

@apply bg-gray-200 hover:bg-gray-300 text-gray-900 font-medium py-2 px-4 rounded-lg transition-colors;

}

.form-input {

@apply mt-1 block w-full rounded-md border-gray-300 shadow-sm focus:border-primary-500 focus:ring-primary-500 sm:text-sm;

}

.card {

@apply bg-white shadow-md rounded-lg p-6;

}

}

**Project Structure**

Create this folder structure in src/:

src/

├── components/

│ ├── auth/

│ ├── common/

│ ├── admin/

│ ├── store/

│ ├── storeOwner/

│ └── user/

├── context/

├── services/

├── utils/

├── pages/

└── App.js

**2. Core Services and Context**

**API Service (src/services/api.js)**

import axios from 'axios';

const API\_BASE\_URL = process.env.REACT\_APP\_API\_URL || 'http://localhost:5000/api';

const api = axios.create({

baseURL: API\_BASE\_URL,

headers: { 'Content-Type': 'application/json' },

});

// Request interceptor to add auth token

api.interceptors.request.use((config) => {

const token = localStorage.getItem('token');

if (token) {

config.headers.Authorization = `Bearer ${token}`;

}

return config;

});

// Response interceptor for error handling

api.interceptors.response.use(

(response) => response,

(error) => {

if (error.response?.status === 401) {

localStorage.removeItem('token');

localStorage.removeItem('user');

window.location.href = '/login';

}

return Promise.reject(error);

}

);

// Auth API calls

export const authAPI = {

login: (credentials) => api.post('/auth/login', credentials),

register: (userData) => api.post('/auth/register', userData),

updatePassword: (passwordData) => api.put('/auth/update-password', passwordData),

};

// Store API calls

export const storeAPI = {

getAllStores: (params) => api.get('/stores', { params }),

getStoreById: (id) => api.get(`/stores/${id}`),

createStore: (storeData) => api.post('/stores', storeData),

updateStore: (id, storeData) => api.put(`/stores/${id}`, storeData),

deleteStore: (id) => api.delete(`/stores/${id}`),

};

// Rating API calls

export const ratingAPI = {

submitRating: (ratingData) => api.post('/ratings', ratingData),

getUserRatings: (params) => api.get('/ratings/user', { params }),

getStoreRatings: (storeId, params) => api.get(`/ratings/store/${storeId}`, { params }),

deleteRating: (id) => api.delete(`/ratings/${id}`),

};

// Admin API calls

export const adminAPI = {

getDashboardStats: () => api.get('/admin/dashboard'),

getAllUsers: (params) => api.get('/admin/users', { params }),

createUser: (userData) => api.post('/admin/users', userData),

updateUser: (id, userData) => api.put(`/admin/users/${id}`, userData),

deleteUser: (id) => api.delete(`/admin/users/${id}`),

};

// Store Owner API calls

export const storeOwnerAPI = {

getOwnerStore: () => api.get('/store-owner/store'),

getStoreRatings: (params) => api.get('/store-owner/store/ratings', { params }),

};

export default api;

**Authentication Context (src/context/AuthContext.js)**

import React, { createContext, useContext, useReducer, useEffect } from 'react';

import { authAPI } from '../services/api';

const AuthContext = createContext();

const authReducer = (state, action) => {

switch (action.type) {

case 'LOGIN\_START':

return { ...state, loading: true, error: null };

case 'LOGIN\_SUCCESS':

return {

...state,

loading: false,

isAuthenticated: true,

user: action.payload.user,

token: action.payload.token,

error: null

};

case 'LOGIN\_FAILURE':

return {

...state,

loading: false,

isAuthenticated: false,

user: null,

token: null,

error: action.payload

};

case 'LOGOUT':

return {

...state,

isAuthenticated: false,

user: null,

token: null,

loading: false,

error: null

};

case 'SET\_LOADING':

return { ...state, loading: action.payload };

case 'CLEAR\_ERROR':

return { ...state, error: null };

default:

return state;

}

};

const initialState = {

isAuthenticated: false,

user: null,

token: null,

loading: true,

error: null,

};

export const AuthProvider = ({ children }) => {

const [state, dispatch] = useReducer(authReducer, initialState);

useEffect(() => {

const token = localStorage.getItem('token');

const user = localStorage.getItem('user');

if (token && user) {

dispatch({

type: 'LOGIN\_SUCCESS',

payload: { token, user: JSON.parse(user) },

});

} else {

dispatch({ type: 'SET\_LOADING', payload: false });

}

}, []);

const login = async (credentials) => {

try {

dispatch({ type: 'LOGIN\_START' });

const response = await authAPI.login(credentials);

const { token, user } = response.data;

localStorage.setItem('token', token);

localStorage.setItem('user', JSON.stringify(user));

dispatch({ type: 'LOGIN\_SUCCESS', payload: { token, user } });

return { success: true };

} catch (error) {

const errorMessage = error.response?.data?.error || 'Login failed';

dispatch({ type: 'LOGIN\_FAILURE', payload: errorMessage });

return { success: false, error: errorMessage };

}

};

const register = async (userData) => {

try {

dispatch({ type: 'LOGIN\_START' });

const response = await authAPI.register(userData);

const { token, user } = response.data;

localStorage.setItem('token', token);

localStorage.setItem('user', JSON.stringify(user));

dispatch({ type: 'LOGIN\_SUCCESS', payload: { token, user } });

return { success: true };

} catch (error) {

const errorMessage = error.response?.data?.error || 'Registration failed';

dispatch({ type: 'LOGIN\_FAILURE', payload: errorMessage });

return { success: false, error: errorMessage };

}

};

const logout = () => {

localStorage.removeItem('token');

localStorage.removeItem('user');

dispatch({ type: 'LOGOUT' });

};

const clearError = () => {

dispatch({ type: 'CLEAR\_ERROR' });

};

return (

<AuthContext.Provider value={{ ...state, login, register, logout, clearError }}>

{children}

</AuthContext.Provider>

);

};

export const useAuth = () => {

const context = useContext(AuthContext);

if (!context) {

throw new Error('useAuth must be used within an AuthProvider');

}

return context;

};

**Utilities (src/utils/constants.js)**

export const USER\_ROLES = {

SYSTEM\_ADMIN: 'system\_admin',

NORMAL\_USER: 'normal\_user',

STORE\_OWNER: 'store\_owner',

};

export const RATING\_LABELS = {

1: 'Poor',

2: 'Fair',

3: 'Good',

4: 'Very Good',

5: 'Excellent',

};

**Validation (src/utils/validation.js)**

export const validateName = (name) => {

if (!name) return 'Name is required';

if (name.length < 20) return 'Name must be at least 20 characters';

if (name.length > 60) return 'Name must not exceed 60 characters';

return '';

};

export const validateEmail = (email) => {

if (!email) return 'Email is required';

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

if (!emailRegex.test(email)) return 'Please enter a valid email address';

return '';

};

export const validatePassword = (password) => {

if (!password) return 'Password is required';

if (password.length < 8) return 'Password must be at least 8 characters';

if (password.length > 16) return 'Password must not exceed 16 characters';

if (!/[A-Z]/.test(password)) return 'Password must contain at least one uppercase letter';

if (!/[!@#$%^&\*(),.?":{}|<>]/.test(password)) return 'Password must contain at least one special character';

return '';

};

export const validateAddress = (address) => {

if (address && address.length > 400) return 'Address must not exceed 400 characters';

return '';

};

export const validateRating = (rating) => {

const num = parseInt(rating);

if (isNaN(num) || num < 1 || num > 5) return 'Rating must be between 1 and 5';

return '';

};

**3. Authentication System**

**Protected Route (src/components/auth/ProtectedRoute.js)**

import React from 'react';

import { Navigate, useLocation } from 'react-router-dom';

import { useAuth } from '../../context/AuthContext';

const ProtectedRoute = ({ children, allowedRoles = [] }) => {

const { isAuthenticated, user, loading } = useAuth();

const location = useLocation();

if (loading) {

return (

<div className="min-h-screen flex items-center justify-center">

<div className="animate-spin rounded-full h-12 w-12 border-b-2 border-primary-600"></div>

</div>

);

}

if (!isAuthenticated) {

return <Navigate to="/login" state={{ from: location }} replace />;

}

if (allowedRoles.length > 0 && !allowedRoles.includes(user?.role)) {

return <Navigate to="/unauthorized" replace />;

}

return children;

};

export default ProtectedRoute;

**Login Form (src/components/auth/LoginForm.js)**

import React, { useState } from 'react';

import { Link, useNavigate, useLocation } from 'react-router-dom';

import { useAuth } from '../../context/AuthContext';

import { validateEmail, validatePassword } from '../../utils/validation';

import { Eye, EyeOff, LogIn } from 'lucide-react';

const LoginForm = () => {

const [formData, setFormData] = useState({ email: '', password: '' });

const [errors, setErrors] = useState({});

const [showPassword, setShowPassword] = useState(false);

const { login, loading, error, clearError } = useAuth();

const navigate = useNavigate();

const location = useLocation();

const handleChange = (e) => {

const { name, value } = e.target;

setFormData(prev => ({ ...prev, [name]: value }));

if (errors[name]) {

setErrors(prev => ({ ...prev, [name]: '' }));

}

if (error) clearError();

};

const validateForm = () => {

const newErrors = {};

const emailError = validateEmail(formData.email);

if (emailError) newErrors.email = emailError;

const passwordError = validatePassword(formData.password);

if (passwordError) newErrors.password = passwordError;

setErrors(newErrors);

return Object.keys(newErrors).length === 0;

};

const handleSubmit = async (e) => {

e.preventDefault();

if (!validateForm()) return;

const result = await login(formData);

if (result.success) {

const from = location.state?.from?.pathname || '/dashboard';

navigate(from, { replace: true });

}

};

return (

<div className="min-h-screen flex items-center justify-center bg-gray-50 py-12 px-4">

<div className="max-w-md w-full space-y-8">

<div>

<h2 className="mt-6 text-center text-3xl font-extrabold text-gray-900">

Sign in to your account

</h2>

<p className="mt-2 text-center text-sm text-gray-600">

Or{' '}

<Link to="/register" className="font-medium text-primary-600 hover:text-primary-500">

create a new account

</Link>

</p>

</div>

<form className="mt-8 space-y-6" onSubmit={handleSubmit}>

{error && (

<div className="bg-red-50 border border-red-200 text-red-600 px-4 py-3 rounded-md text-sm">

{error}

</div>

)}

<div className="space-y-4">

<div>

<label htmlFor="email" className="block text-sm font-medium text-gray-700">

Email address

</label>

<input

id="email"

name="email"

type="email"

required

className={`form-input ${errors.email ? 'border-red-300' : ''}`}

placeholder="Enter your email"

value={formData.email}

onChange={handleChange}

/>

{errors.email && <p className="mt-1 text-sm text-red-600">{errors.email}</p>}

</div>

<div>

<label htmlFor="password" className="block text-sm font-medium text-gray-700">

Password

</label>

<div className="relative">

<input

id="password"

name="password"

type={showPassword ? 'text' : 'password'}

required

className={`form-input pr-10 ${errors.password ? 'border-red-300' : ''}`}

placeholder="Enter your password"

value={formData.password}

onChange={handleChange}

/>

<button

type="button"

className="absolute inset-y-0 right-0 pr-3 flex items-center"

onClick={() => setShowPassword(!showPassword)}

>

{showPassword ? <EyeOff className="h-5 w-5 text-gray-400" /> : <Eye className="h-5 w-5 text-gray-400" />}

</button>

</div>

{errors.password && <p className="mt-1 text-sm text-red-600">{errors.password}</p>}

</div>

</div>

<button

type="submit"

disabled={loading}

className="group relative w-full flex justify-center py-3 px-4 border border-transparent text-sm font-medium rounded-md text-white bg-primary-600 hover:bg-primary-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-primary-500 disabled:opacity-50"

>

<LogIn className="h-5 w-5 mr-2" />

{loading ? 'Signing in...' : 'Sign in'}

</button>

</form>

</div>

</div>

);

};

export default LoginForm;

**Register Form (src/components/auth/RegisterForm.js)**

import React, { useState } from 'react';

import { Link, useNavigate } from 'react-router-dom';

import { useAuth } from '../../context/AuthContext';

import { validateName, validateEmail, validatePassword, validateAddress } from '../../utils/validation';

import { Eye, EyeOff, UserPlus } from 'lucide-react';

const RegisterForm = () => {

const [formData, setFormData] = useState({

name: '', email: '', password: '', confirmPassword: '', address: ''

});

const [errors, setErrors] = useState({});

const [showPassword, setShowPassword] = useState(false);

const [showConfirmPassword, setShowConfirmPassword] = useState(false);

const { register, loading, error, clearError } = useAuth();

const navigate = useNavigate();

const handleChange = (e) => {

const { name, value } = e.target;

setFormData(prev => ({ ...prev, [name]: value }));

if (errors[name]) {

setErrors(prev => ({ ...prev, [name]: '' }));

}

if (error) clearError();

};

const validateForm = () => {

const newErrors = {};

const nameError = validateName(formData.name);

if (nameError) newErrors.name = nameError;

const emailError = validateEmail(formData.email);

if (emailError) newErrors.email = emailError;

const passwordError = validatePassword(formData.password);

if (passwordError) newErrors.password = passwordError;

if (formData.password !== formData.confirmPassword) {

newErrors.confirmPassword = 'Passwords do not match';

}

const addressError = validateAddress(formData.address);

if (addressError) newErrors.address = addressError;

setErrors(newErrors);

return Object.keys(newErrors).length === 0;

};

const handleSubmit = async (e) => {

e.preventDefault();

if (!validateForm()) return;

const result = await register({

name: formData.name,

email: formData.email,

password: formData.password,

address: formData.address,

});

if (result.success) {

navigate('/dashboard', { replace: true });

}

};

return (

<div className="min-h-screen flex items-center justify-center bg-gray-50 py-12 px-4">

<div className="max-w-md w-full space-y-8">

<div>

<h2 className="mt-6 text-center text-3xl font-extrabold text-gray-900">

Create your account

</h2>

<p className="mt-2 text-center text-sm text-gray-600">

Or{' '}

<Link to="/login" className="font-medium text-primary-600 hover:text-primary-500">

sign in to existing account

</Link>

</p>

</div>

<form className="mt-8 space-y-6" onSubmit={handleSubmit}>

{error && (

<div className="bg-red-50 border border-red-200 text-red-600 px-4 py-3 rounded-md text-sm">

{error}

</div>

)}

<div className="space-y-4">

<div>

<label htmlFor="name" className="block text-sm font-medium text-gray-700">

Full Name <span className="text-red-500">\*</span>

</label>

<input

id="name"

name="name"

type="text"

required

className={`form-input ${errors.name ? 'border-red-300' : ''}`}

placeholder="Enter your full name (20-60 characters)"

value={formData.name}

onChange={handleChange}

/>

{errors.name && <p className="mt-1 text-sm text-red-600">{errors.name}</p>}

</div>

<div>

<label htmlFor="email" className="block text-sm font-medium text-gray-700">

Email address <span className="text-red-500">\*</span>

</label>

<input

id="email"

name="email"

type="email"

required

className={`form-input ${errors.email ? 'border-red-300' : ''}`}

placeholder="Enter your email"

value={formData.email}

onChange={handleChange}

/>

{errors.email && <p className="mt-1 text-sm text-red-600">{errors.email}</p>}

</div>

<div>

<label htmlFor="password" className="block text-sm font-medium text-gray-700">

Password <span className="text-red-500">\*</span>

</label>

<div className="relative">

<input

id="password"

name="password"

type={showPassword ? 'text' : 'password'}

required

className={`form-input pr-10 ${errors.password ? 'border-red-300' : ''}`}

placeholder="8-16 chars, uppercase & special character"

value={formData.password}

onChange={handleChange}

/>

<button

type="button"

className="absolute inset-y-0 right-0 pr-3 flex items-center"

onClick={() => setShowPassword(!showPassword)}

>

{showPassword ? <EyeOff className="h-5 w-5 text-gray-400" /> : <Eye className="h-5 w-5 text-gray-400" />}

</button>

</div>

{errors.password && <p className="mt-1 text-sm text-red-600">{errors.password}</p>}

</div>

<div>

<label htmlFor="confirmPassword" className="block text-sm font-medium text-gray-700">

Confirm Password <span className="text-red-500">\*</span>

</label>

<div className="relative">

<input

id="confirmPassword"

name="confirmPassword"

type={showConfirmPassword ? 'text' : 'password'}

required

className={`form-input pr-10 ${errors.confirmPassword ? 'border-red-300' : ''}`}

placeholder="Confirm your password"

value={formData.confirmPassword}

onChange={handleChange}

/>

<button

type="button"

className="absolute inset-y-0 right-0 pr-3 flex items-center"

onClick={() => setShowConfirmPassword(!showConfirmPassword)}

>

{showConfirmPassword ? <EyeOff className="h-5 w-5 text-gray-400" /> : <Eye className="h-5 w-5 text-gray-400" />}

</button>

</div>

{errors.confirmPassword && <p className="mt-1 text-sm text-red-600">{errors.confirmPassword}</p>}

</div>

<div>

<label htmlFor="address" className="block text-sm font-medium text-gray-700">

Address

</label>

<textarea

id="address"

name="address"

rows="3"

className={`form-input ${errors.address ? 'border-red-300' : ''}`}

placeholder="Enter your address (max 400 characters)"

value={formData.address}

onChange={handleChange}

/>

{errors.address && <p className="mt-1 text-sm text-red-600">{errors.address}</p>}

<p className="mt-1 text-sm text-gray-500">{formData.address.length}/400 characters</p>

</div>

</div>

<button

type="submit"

disabled={loading}

className="group relative w-full flex justify-center py-3 px-4 border border-transparent text-sm font-medium rounded-md text-white bg-primary-600 hover:bg-primary-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-primary-500 disabled:opacity-50"

>

<UserPlus className="h-5 w-5 mr-2" />

{loading ? 'Creating account...' : 'Create account'}

</button>

</form>

</div>

</div>

);

};

export default RegisterForm;

**4. Common Components**

**Layout Component (src/components/common/Layout.js)**

import React from 'react';

import Header from './Header';

import Sidebar from './Sidebar';

import { useAuth } from '../../context/AuthContext';

const Layout = ({ children }) => {

const { isAuthenticated } = useAuth();

if (!isAuthenticated) {

return children;

}

return (

<div className="flex h-screen bg-gray-100">

<Sidebar />

<div className="flex-1 flex flex-col overflow-hidden">

<Header />

<main className="flex-1 overflow-x-hidden overflow-y-auto bg-gray-100">

<div className="container mx-auto px-6 py-8">

{children}

</div>

</main>

</div>

</div>

);

};

export default Layout;

**Header Component (src/components/common/Header.js)**

import React from 'react';

import { useAuth } from '../../context/AuthContext';

import { useNavigate } from 'react-router-dom';

import { LogOut, User, Settings } from 'lucide-react';

import { USER\_ROLES } from '../../utils/constants';

const Header = () => {

const { user, logout } = useAuth();

const navigate = useNavigate();

const handleLogout = () => {

logout();

navigate('/login');

};

const getRoleDisplayName = (role) => {

switch (role) {

case USER\_ROLES.SYSTEM\_ADMIN: return 'System Admin';

case USER\_ROLES.STORE\_OWNER: return 'Store Owner';

case USER\_ROLES.NORMAL\_USER: return 'User';

default: return 'User';

}

};

return (

<header className="bg-white shadow-sm border-b">

<div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">

<div className="flex justify-between items-center h-16">

<div className="flex items-center">

<h1 className="text-xl font-semibold text-gray-900">

Store Rating System

</h1>

</div>

<div className="flex items-center space-x-4">

<div className="flex items-center space-x-2">

<User className="h-5 w-5 text-gray-400" />

<div className="text-sm">

<div className="font-medium text-gray-900">{user?.name}</div>

<div className="text-gray-500">{getRoleDisplayName(user?.role)}</div>

</div>

</div>

<div className="flex items-center space-x-2">

<button

onClick={() => navigate('/profile')}

className="p-2 text-gray-400 hover:text-gray-500 rounded-full hover:bg-gray-100"

title="Profile Settings"

>

<Settings className="h-5 w-5" />

</button>

<button

onClick={handleLogout}

className="p-2 text-gray-400 hover:text-red-500 rounded-full hover:bg-red-50"

title="Logout"

>

<LogOut className="h-5 w-5" />

</button>

</div>

</div>

</div>

</div>

</header>

);

};

export default Header;

**Sidebar Component (src/components/common/Sidebar.js)**

import React from 'react';

import { NavLink } from 'react-router-dom';

import { useAuth } from '../../context/AuthContext';

import { Home, Store, Star, Users, BarChart3, ShoppingBag, UserCheck } from 'lucide-react';

import { USER\_ROLES } from '../../utils/constants';

const Sidebar = () => {

const { user } = useAuth();

const getNavigationItems = () => {

const items = [

{ name: 'Dashboard', href: '/dashboard', icon: Home }

];

if (user?.role === USER\_ROLES.SYSTEM\_ADMIN) {

items.push(

{ name: 'User Management', href: '/admin/users', icon: Users },

{ name: 'Store Management', href: '/admin/stores', icon: Store },

{ name: 'Analytics', href: '/admin/analytics', icon: BarChart3 }

);

}

if (user?.role === USER\_