E:\expense\_tracker\frontend

E:\expense\_tracker\frontend\node\_modules

E:\expense\_tracker\frontend\public

E:\expense\_tracker\frontend\public\index.html

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <meta name="theme-color" content="#000000" />

    <meta

      name="description"

      content="Web site created using create-react-app"

    />

    <link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />

    <!--

      manifest.json provides metadata used when your web app is installed on a

      user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/

    -->

    <link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />

    <!--

      Notice the use of %PUBLIC\_URL% in the tags above.

      It will be replaced with the URL of the `public` folder during the build.

      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC\_URL%/favicon.ico" will

      work correctly both with client-side routing and a non-root public URL.

      Learn how to configure a non-root public URL by running `npm run build`.

    -->

    <title>React App</title>

  </head>

  <body>

    <noscript>You need to enable JavaScript to run this app.</noscript>

    <div id="root"></div>

    <!--

      This HTML file is a template.

      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.

      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.

      To create a production bundle, use `npm run build` or `yarn build`.

    -->

  </body>

</html>

E:\expense\_tracker\frontend\public\manifest.json

{

  "short\_name": "React App",

  "name": "Create React App Sample",

  "icons": [

    {

      "src": "favicon.ico",

      "sizes": "64x64 32x32 24x24 16x16",

      "type": "image/x-icon"

    },

    {

      "src": "logo192.png",

      "type": "image/png",

      "sizes": "192x192"

    },

    {

      "src": "logo512.png",

      "type": "image/png",

      "sizes": "512x512"

    }

  ],

  "start\_url": ".",

  "display": "standalone",

  "theme\_color": "#000000",

  "background\_color": "#ffffff"

}

E:\expense\_tracker\frontend\public\robots.txt

# https://www.robotstxt.org/robotstxt.html

User-agent: \*

Disallow:

E:\expense\_tracker\frontend\src

E:\expense\_tracker\frontend\src\pages\NotFound.jsx

/\*\*

 \* NotFound Component

 \* 404 Error Page - Page Not Found

 \* Displayed when user navigates to non-existent route

 \*/

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

import {

  MdHome,

  MdArrowBack,

  MdSearch,

  MdSentimentDissatisfied,

} from 'react-icons/md';

const NotFound = () => {

  const navigate = useNavigate();

  return (

    <div className="min-h-screen bg-gradient-to-br from-blue-50 via-purple-50 to-pink-50 flex items-center justify-center p-6">

      <div className="max-w-4xl w-full">

        <div className="text-center">

          {/\* 404 Illustration \*/}

          <div className="mb-8">

            <div className="relative inline-block">

              {/\* Large 404 Text \*/}

              <h1 className="text-[200px] md:text-[280px] font-extrabold text-transparent bg-clip-text bg-gradient-to-r from-blue-400 via-purple-500 to-pink-500 leading-none select-none">

                404

              </h1>

              {/\* Sad Face Icon \*/}

              <div className="absolute top-1/2 left-1/2 transform -translate-x-1/2 -translate-y-1/2">

                <MdSentimentDissatisfied className="text-6xl md:text-8xl text-gray-400 animate-bounce" />

              </div>

            </div>

          </div>

          {/\* Error Message \*/}

          <div className="mb-8">

            <h2 className="text-3xl md:text-4xl font-bold text-gray-900 mb-4">

              Oops! Page Not Found

            </h2>

            <p className="text-lg text-gray-600 mb-2">

              The page you're looking for doesn't exist or has been moved.

            </p>

            <p className="text-gray-500">

              Let's get you back on track!

            </p>

          </div>

          {/\* Search Suggestions \*/}

          <div className="bg-white rounded-2xl shadow-xl p-6 md:p-8 mb-8 max-w-2xl mx-auto">

            <div className="flex items-center gap-3 mb-4">

              <MdSearch className="text-2xl text-gray-400" />

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

                Looking for something?

              </h3>

            </div>

            <div className="grid grid-cols-1 md:grid-cols-2 gap-3 text-left">

              <Link

                to="/dashboard"

                className="p-4 border border-gray-200 rounded-lg hover:border-blue-400 hover:bg-blue-50 transition-all group"

              >

                <h4 className="font-semibold text-gray-900 group-hover:text-blue-600 mb-1">

                  Dashboard

                </h4>

                <p className="text-sm text-gray-500">

                  View your expense overview

                </p>

              </Link>

              <Link

                to="/expenses"

                className="p-4 border border-gray-200 rounded-lg hover:border-purple-400 hover:bg-purple-50 transition-all group"

              >

                <h4 className="font-semibold text-gray-900 group-hover:text-purple-600 mb-1">

                  Expenses

                </h4>

                <p className="text-sm text-gray-500">

                  Manage your expenses

                </p>

              </Link>

              <Link

                to="/analytics"

                className="p-4 border border-gray-200 rounded-lg hover:border-green-400 hover:bg-green-50 transition-all group"

              >

                <h4 className="font-semibold text-gray-900 group-hover:text-green-600 mb-1">

                  Analytics

                </h4>

                <p className="text-sm text-gray-500">

                  View spending insights

                </p>

              </Link>

              <Link

                to="/ai-insights"

                className="p-4 border border-gray-200 rounded-lg hover:border-pink-400 hover:bg-pink-50 transition-all group"

              >

                <h4 className="font-semibold text-gray-900 group-hover:text-pink-600 mb-1">

                  AI Insights

                </h4>

                <p className="text-sm text-gray-500">

                  Get AI-powered advice

                </p>

              </Link>

            </div>

          </div>

          {/\* Action Buttons \*/}

          <div className="flex flex-col sm:flex-row gap-4 justify-center items-center">

            <button

              onClick={() => navigate(-1)}

              className="flex items-center gap-2 px-8 py-4 bg-white text-gray-700 rounded-xl font-semibold shadow-lg hover:shadow-xl hover:bg-gray-50 transition-all transform hover:-translate-y-1"

            >

              <MdArrowBack className="text-xl" />

              Go Back

            </button>

            <Link

              to="/dashboard"

              className="flex items-center gap-2 px-8 py-4 bg-gradient-to-r from-blue-600 to-purple-600 text-white rounded-xl font-semibold shadow-lg hover:shadow-xl hover:from-blue-700 hover:to-purple-700 transition-all transform hover:-translate-y-1"

            >

              <MdHome className="text-xl" />

              Go to Dashboard

            </Link>

          </div>

          {/\* Helpful Tip \*/}

          <div className="mt-12 p-4 bg-blue-50 border border-blue-200 rounded-lg max-w-2xl mx-auto">

            <p className="text-sm text-gray-600">

              💡 <strong className="text-gray-900">Helpful Tip:</strong> If you believe this is an error,

              please check the URL or contact support.

            </p>

          </div>

        </div>

      </div>

    </div>

  );

};

export default NotFound;

E:\expense\_tracker\frontend\src\pages\Unauthorized.jsx

/\*\*

 \* Unauthorized Component

 \* 401 Error Page - Unauthorized Access

 \* Displayed when user tries to access protected routes without authentication

 \*/

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

import {

  MdLock,

  MdLogin,

  MdHome,

  MdArrowBack,

  MdSecurity,

  MdWarning,

} from 'react-icons/md';

import { isAuthenticated } from '../utils/auth';

const Unauthorized = () => {

  const navigate = useNavigate();

  const location = useLocation();

  const isLoggedIn = isAuthenticated();

  return (

    <div className="min-h-screen bg-gradient-to-br from-red-50 via-orange-50 to-yellow-50 flex items-center justify-center p-6">

      <div className="max-w-4xl w-full">

        <div className="text-center">

          {/\* 401 Illustration \*/}

          <div className="mb-8">

            <div className="relative inline-block">

              {/\* Large 401 Text \*/}

              <h1 className="text-[180px] md:text-[250px] font-extrabold text-transparent bg-clip-text bg-gradient-to-r from-red-400 via-orange-500 to-yellow-500 leading-none select-none">

                401

              </h1>

              {/\* Lock Icon \*/}

              <div className="absolute top-1/2 left-1/2 transform -translate-x-1/2 -translate-y-1/2">

                <div className="relative">

                  <MdLock className="text-6xl md:text-8xl text-gray-400" />

                  <div className="absolute -top-2 -right-2 w-8 h-8 bg-red-500 rounded-full flex items-center justify-center animate-pulse">

                    <MdWarning className="text-white text-lg" />

                  </div>

                </div>

              </div>

            </div>

          </div>

          {/\* Error Message \*/}

          <div className="mb-8">

            <h2 className="text-3xl md:text-4xl font-bold text-gray-900 mb-4">

              Access Denied

            </h2>

            <p className="text-lg text-gray-600 mb-2">

              {isLoggedIn

                ? "You don't have permission to access this page."

                : "You need to be logged in to access this page."

              }

            </p>

            <p className="text-gray-500">

              {isLoggedIn

                ? "Please contact an administrator if you believe this is an error."

                : "Please sign in with your account credentials."}

            </p>

          </div>

          {/\* Info Card \*/}

          <div className="bg-white rounded-2xl shadow-xl p-6 md:p-8 mb-8 max-w-2xl mx-auto">

            <div className="flex items-start gap-4 mb-6">

              <div className="flex-shrink-0 w-12 h-12 bg-red-100 rounded-full flex items-center justify-center">

                <MdSecurity className="text-2xl text-red-600" />

              </div>

              <div className="flex-1 text-left">

                <h3 className="text-lg font-semibold text-gray-900 mb-2">

                  Why am I seeing this?

                </h3>

                <ul className="space-y-2 text-sm text-gray-600">

                  {!isLoggedIn ? (

                    <>

                      <li className="flex items-start gap-2">

                        <span className="text-red-500 mt-0.5">•</span>

                        <span>You're not currently logged in to the application</span>

                      </li>

                      <li className="flex items-start gap-2">

                        <span className="text-red-500 mt-0.5">•</span>

                        <span>Your session may have expired</span>

                      </li>

                      <li className="flex items-start gap-2">

                        <span className="text-red-500 mt-0.5">•</span>

                        <span>You've been logged out for security reasons</span>

                      </li>

                    </>

                  ) : (

                    <>

                      <li className="flex items-start gap-2">

                        <span className="text-red-500 mt-0.5">•</span>

                        <span>This page requires special permissions</span>

                      </li>

                      <li className="flex items-start gap-2">

                        <span className="text-red-500 mt-0.5">•</span>

                        <span>Your account doesn't have access to this resource</span>

                      </li>

                      <li className="flex items-start gap-2">

                        <span className="text-red-500 mt-0.5">•</span>

                        <span>The page URL might be incorrect</span>

                      </li>

                    </>

                  )}

                </ul>

              </div>

            </div>

            {/\* Security Notice \*/}

            <div className="bg-yellow-50 border border-yellow-200 rounded-lg p-4">

              <div className="flex items-start gap-3">

                <MdWarning className="text-xl text-yellow-600 flex-shrink-0 mt-0.5" />

                <p className="text-sm text-gray-700 text-left">

                  <strong className="text-gray-900">Security Notice:</strong> If you're

                  attempting to access this page without authorization, please note that all

                  access attempts are logged and monitored.

                </p>

              </div>

            </div>

          </div>

          {/\* Action Buttons \*/}

          <div className="flex flex-col sm:flex-row gap-4 justify-center items-center mb-8">

            {!isLoggedIn ? (

              <>

                <Link

                  to="/login"

                  state={{ from: location }}

                  className="flex items-center gap-2 px-8 py-4 bg-gradient-to-r from-red-600 to-orange-600 text-white rounded-xl font-semibold shadow-lg hover:shadow-xl hover:from-red-700 hover:to-orange-700 transition-all transform hover:-translate-y-1"

                >

                  <MdLogin className="text-xl" />

                  Sign In

                </Link>

                <Link

                  to="/register"

                  className="flex items-center gap-2 px-8 py-4 bg-white text-gray-700 rounded-xl font-semibold shadow-lg hover:shadow-xl hover:bg-gray-50 transition-all transform hover:-translate-y-1"

                >

                  Create Account

                </Link>

              </>

            ) : (

              <>

                <button

                  onClick={() => navigate(-1)}

                  className="flex items-center gap-2 px-8 py-4 bg-white text-gray-700 rounded-xl font-semibold shadow-lg hover:shadow-xl hover:bg-gray-50 transition-all transform hover:-translate-y-1"

                >

                  <MdArrowBack className="text-xl" />

                  Go Back

                </button>

                <Link

                  to="/dashboard"

                  className="flex items-center gap-2 px-8 py-4 bg-gradient-to-r from-red-600 to-orange-600 text-white rounded-xl font-semibold shadow-lg hover:shadow-xl hover:from-red-700 hover:to-orange-700 transition-all transform hover:-translate-y-1"

                >

                  <MdHome className="text-xl" />

                  Go to Dashboard

                </Link>

              </>

            )}

          </div>

          {/\* Quick Access Links \*/}

          {isLoggedIn && (

            <div className="mt-8">

              <p className="text-sm text-gray-600 mb-4">Quick access to your pages:</p>

              <div className="flex flex-wrap gap-3 justify-center">

                <Link

                  to="/expenses"

                  className="px-4 py-2 bg-white text-gray-700 rounded-lg text-sm font-medium hover:bg-gray-50 transition-colors shadow"

                >

                  My Expenses

                </Link>

                <Link

                  to="/analytics"

                  className="px-4 py-2 bg-white text-gray-700 rounded-lg text-sm font-medium hover:bg-gray-50 transition-colors shadow"

                >

                  Analytics

                </Link>

                <Link

                  to="/profile"

                  className="px-4 py-2 bg-white text-gray-700 rounded-lg text-sm font-medium hover:bg-gray-50 transition-colors shadow"

                >

                  Profile

                </Link>

              </div>

            </div>

          )}

        </div>

      </div>

    </div>

  );

};

export default Unauthorized;

E:\expense\_tracker\frontend\src\pages\ai

E:\expense\_tracker\frontend\src\pages\ai\AiInsights.jsx

/\*\*

 \* AiInsights Component

 \* AI-powered expense analysis using OpenAI

 \* Backend: POST /api/ai/query, GET /api/ai/suggestions

 \*/

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

import { toast } from 'react-hot-toast';

import {

  MdPsychology,

  MdSend,

  MdLightbulb,

  MdRefresh,

  MdHistory,

  MdTrendingUp,

  MdWarning,

} from 'react-icons/md';

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

const AiInsights = () => {

  const [query, setQuery] = useState('');

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

  const [suggestionsLoading, setSuggestionsLoading] = useState(false);

  const [conversation, setConversation] = useState([]);

  const [suggestions, setSuggestions] = useState(null);

  const [lastSuggestionTime, setLastSuggestionTime] = useState(null);

  const [aiAvailable, setAiAvailable] = useState(true);

  const conversationEndRef = useRef(null);

  // Quick query templates

  const quickQueries = [

    'What were my total expenses this week?',

    'How much did I spend on food this month?',

    'Show me my travel expenses yesterday',

    'What is my biggest spending category?',

    'How much did I spend on GPay today?',

    'Show my cash expenses this week',

    'What were my expenses yesterday?',

    'Compare my spending this month vs last month',

  ];

  // Fetch AI suggestions on component mount

  useEffect(() => {

    fetchSuggestions();

  }, []);

  useEffect(() => {

    // Scroll to latest message when conversation updates

    if (conversationEndRef.current) {

      conversationEndRef.current.scrollIntoView({ behavior: 'smooth' });

    }

  }, [conversation]);

  const fetchSuggestions = async () => {

    try {

      setSuggestionsLoading(true);

      const response = await aiAPI.getSuggestions();

      console.log('✅ Suggestions response:', response.data);

      const suggestionsData = response?.data?.suggestions;

      const isAiGenerated = response?.data?.is\_ai\_generated;

      let suggestionsText = '';

      // Better handling of different response formats

      if (typeof suggestionsData === 'string') {

        suggestionsText = suggestionsData;

      } else if (Array.isArray(suggestionsData)) {

        suggestionsText = suggestionsData.join('\n\n');

      } else {

        suggestionsText = '📊 Track your expenses regularly for better insights';

      }

      setSuggestions({

        text: suggestionsText,

        period: response?.data?.period || 'N/A',

        totalAnalyzed: response?.data?.total\_analyzed || 0,

        isAiGenerated: isAiGenerated !== false,

        note: response?.data?.note || null,

      });

      setLastSuggestionTime(new Date().toISOString());

      // Check if AI is available

      if (response?.data?.note && response.data.note.includes('unavailable')) {

        setAiAvailable(false);

      } else {

        setAiAvailable(true);

      }

    } catch (error) {

      console.error('❌ Error fetching suggestions:', error);

      // Fallback suggestions

      setSuggestions({

        text: '📊 Track your daily expenses to identify spending patterns\n\n💰 Set a monthly budget for each category to stay on track\n\n🔍 Review your top spending categories regularly\n\n📈 Compare your spending month-over-month to spot trends',

        period: 'General Tips',

        totalAnalyzed: 0,

        isAiGenerated: false,

        note: 'AI temporarily unavailable',

      });

      setLastSuggestionTime(new Date().toISOString());

      setAiAvailable(false);

    } finally {

      setSuggestionsLoading(false);

    }

  };

  const handleQuerySubmit = async (e, queryToSubmit = null) => {

    if (e) e.preventDefault();

    const userQuery = (queryToSubmit || query).trim();

    if (!userQuery || loading) {

      if (!userQuery) {

        toast.error('Please enter a question');

      }

      return;

    }

    setQuery('');

    setLoading(true);

    // Add user message to conversation

    const userMessage = {

      type: 'user',

      text: userQuery,

      timestamp: new Date().toISOString(),

    };

    setConversation((prev) => [...prev, userMessage]);

    try {

      console.log('🤖 Sending query:', userQuery);

      const response = await aiAPI.query({ query: userQuery });

      console.log('✅ AI response:', response.data);

      const aiMessage = {

        type: 'ai',

        text: response?.data?.answer || 'No answer received.',

        timestamp: new Date().toISOString(),

        context: response?.data?.context,

      };

      setConversation((prev) => [...prev, aiMessage]);

      setAiAvailable(true);

    } catch (error) {

      console.error('❌ Error querying AI:', error);

      let errorMessage = 'Failed to get AI response. Please try again.';

      let errorDetails = null;

      // Better error handling based on status codes

      if (error.response?.status === 503) {

        errorMessage = error.response?.data?.error || 'AI service is not configured. Please contact the administrator.';

        errorDetails = error.response?.data?.help;

        setAiAvailable(false);

      } else if (error.response?.status === 500) {

        errorMessage = error.response?.data?.fallback || 'AI service is temporarily unavailable. Please try again later.';

        errorDetails = error.response?.data?.error;

      } else if (error.response?.status === 400) {

        errorMessage = error.response?.data?.error || 'Invalid query format. Please rephrase your question.';

      } else if (!error.response) {

        errorMessage = 'Cannot connect to server. Please check your internet connection.';

      }

      const errorAiMessage = {

        type: 'ai',

        text: errorMessage,

        timestamp: new Date().toISOString(),

        isError: true,

        errorDetails: errorDetails,

      };

      setConversation((prev) => [...prev, errorAiMessage]);

      toast.error(errorMessage);

    } finally {

      setLoading(false);

    }

  };

  const handleQuickQuery = (quickQuery) => {

    if (loading) return;

    setQuery(quickQuery);

    setTimeout(() => {

      handleQuerySubmit(null, quickQuery);

    }, 50);

  };

  const handleClearConversation = () => {

    setConversation([]);

    toast.success('Conversation cleared');

  };

  const formatTimestamp = (timestamp) => {

    const date = new Date(timestamp);

    return date.toLocaleString('en-US', {

      month: 'short',

      day: 'numeric',

      hour: '2-digit',

      minute: '2-digit',

    });

  };

  return (

    <div className="min-h-screen bg-gray-50 p-6">

      <div className="max-w-7xl mx-auto">

        {/\* Header \*/}

        <div className="mb-6">

          <div className="flex items-center gap-3 mb-2">

            <div className="w-12 h-12 bg-gradient-to-br from-purple-500 to-pink-500 rounded-xl flex items-center justify-center shadow-lg">

              <MdPsychology className="text-3xl text-white" />

            </div>

            <div>

              <h1 className="text-3xl font-bold text-gray-900">AI Insights</h1>

              <p className="text-sm text-gray-600">

                Ask questions about your expenses in natural language

              </p>

            </div>

          </div>

          {/\* ✅ CHANGED: AI availability warning - updated text \*/}

          {!aiAvailable && (

            <div className="mt-4 bg-orange-50 border border-orange-200 rounded-lg p-4 flex items-start gap-3">

              <MdWarning className="text-orange-500 text-xl flex-shrink-0 mt-0.5" />

              <div>

                <p className="text-sm font-medium text-orange-900">AI Features Limited</p>

                <p className="text-xs text-orange-700 mt-1">

                  OpenAI is not configured. You'll see general tips instead of personalized insights.

                </p>

              </div>

            </div>

          )}

        </div>

        <div className="grid grid-cols-1 lg:grid-cols-3 gap-6">

          {/\* Main Chat Section \*/}

          <div className="lg:col-span-2 space-y-6">

            {/\* AI Suggestions Card \*/}

            <div className="bg-gradient-to-br from-blue-500 to-purple-600 rounded-xl shadow-lg p-6 text-white">

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

                <div className="flex items-center gap-3">

                  <MdLightbulb className="text-3xl" />

                  <div>

                    <h2 className="text-xl font-bold">

                      {suggestions?.isAiGenerated ? 'AI Suggestions' : 'General Tips'}

                    </h2>

                    {suggestions?.note && (

                      <p className="text-xs text-blue-100 mt-1">{suggestions.note}</p>

                    )}

                  </div>

                </div>

                <button

                  onClick={fetchSuggestions}

                  disabled={suggestionsLoading}

                  className="p-2 bg-white bg-opacity-20 rounded-lg hover:bg-opacity-30 transition-colors disabled:opacity-50 flex items-center gap-2"

                  title="Refresh suggestions"

                >

                  <MdRefresh

                    className={`text-xl ${suggestionsLoading ? 'animate-spin' : ''}`}

                  />

                </button>

              </div>

              {suggestionsLoading ? (

                <div className="flex items-center justify-center py-8">

                  <div className="animate-spin rounded-full h-8 w-8 border-b-2 border-white"></div>

                </div>

              ) : suggestions ? (

                <div>

                  <div className="bg-white bg-opacity-10 rounded-lg p-4 mb-3">

                    <div className="prose prose-sm prose-invert max-w-none">

                      {suggestions.text && typeof suggestions.text === 'string' ? (

                        suggestions.text.split('\n').map((line, index) => (

                          line.trim() && (

                            <p key={index} className="mb-2 last:mb-0">

                              {line}

                            </p>

                          )

                        ))

                      ) : (

                        <p>No suggestions available</p>

                      )}

                    </div>

                  </div>

                  <div className="flex items-center justify-between text-sm text-blue-100">

                    <span> Period: {suggestions.period}</span>

                    {suggestions.totalAnalyzed > 0 && (

                      <span>

                        💰 Analyzed: ₹{suggestions.totalAnalyzed.toFixed(2)}

                      </span>

                    )}

                  </div>

                  {lastSuggestionTime && (

                    <div className="text-xs mt-2 text-blue-200">

                      Last updated: {formatTimestamp(lastSuggestionTime)}

                    </div>

                  )}

                </div>

              ) : (

                <p className="text-blue-100">No suggestions available</p>

              )}

            </div>

            {/\* Chat Interface \*/}

            <div className="bg-white rounded-xl shadow-lg overflow-hidden flex flex-col h-[600px]">

              {/\* Chat Header \*/}

              <div className="bg-gradient-to-r from-purple-600 to-pink-600 p-4 flex items-center justify-between">

                <div className="flex items-center gap-3">

                  <div className="w-10 h-10 bg-white bg-opacity-20 rounded-full flex items-center justify-center">

                    <MdPsychology className="text-xl text-white" />

                  </div>

                  <div>

                    <h3 className="font-bold text-white">AI Assistant</h3>

                    {/\* ✅ CHANGED: Updated powered by text \*/}

                    <p className="text-xs text-purple-100">

                      {aiAvailable ? 'Powered by OpenAI' : 'Limited Mode'}

                    </p>

                  </div>

                </div>

                {conversation.length > 0 && (

                  <button

                    onClick={handleClearConversation}

                    className="px-3 py-1 bg-white bg-opacity-20 rounded-lg text-white text-sm hover:bg-opacity-30 transition-colors"

                  >

                    Clear

                  </button>

                )}

              </div>

              {/\* Conversation Area \*/}

              <div className="flex-1 overflow-y-auto p-4 space-y-4 bg-gray-50">

                {conversation.length === 0 ? (

                  <div className="flex flex-col items-center justify-center h-full text-center px-4">

                    <MdPsychology className="text-6xl text-gray-300 mb-4" />

                    <h3 className="text-lg font-bold text-gray-900 mb-2">

                      Ask me anything about your expenses!

                    </h3>

                    <p className="text-gray-600 mb-4">

                      Try questions like "How much did I spend on food this week?" or "Show my GPay expenses"

                    </p>

                    <p className="text-sm text-gray-500">

                      💡 Use the quick questions on the right to get started

                    </p>

                  </div>

                ) : (

                  conversation.map((message, index) => (

                    <div

                      key={index}

                      className={`flex ${

                        message.type === 'user' ? 'justify-end' : 'justify-start'

                      }`}

                    >

                      <div

                        className={`max-w-[85%] rounded-xl p-4 ${

                          message.type === 'user'

                            ? 'bg-gradient-to-r from-blue-600 to-purple-600 text-white shadow-lg'

                            : message.isError

                            ? 'bg-red-50 text-red-900 border-2 border-red-200'

                            : 'bg-white shadow-md text-gray-900'

                        }`}

                      >

                        <div className="flex items-start gap-3">

                          {message.type === 'ai' && (

                            <div className={`flex-shrink-0 w-8 h-8 rounded-full flex items-center justify-center ${

                              message.isError ? 'bg-red-500' : 'bg-gradient-to-br from-purple-500 to-pink-500'

                            }`}>

                              {message.isError ? (

                                <MdWarning className="text-white text-sm" />

                              ) : (

                                <MdPsychology className="text-white text-sm" />

                              )}

                            </div>

                          )}

                          <div className="flex-1 min-w-0">

                            <div className="prose prose-sm max-w-none">

                              {message.text.split('\n').map((line, idx) => (

                                line.trim() && (

                                  <p key={idx} className="mb-2 last:mb-0">

                                    {line}

                                  </p>

                                )

                              ))}

                            </div>

                            {message.errorDetails && (

                              <div className="mt-2 p-2 bg-red-100 rounded text-xs text-red-700">

                                <strong>Details:</strong> {message.errorDetails}

                              </div>

                            )}

                            {message.context && (

                              <div

                                className={`mt-3 pt-3 border-t text-xs ${

                                  message.type === 'user'

                                    ? 'border-blue-400 text-blue-100'

                                    : 'border-gray-200 text-gray-500'

                                }`}

                              >

                                <span>

                                  📅 {message.context.period} •

                                  🔢 {message.context.expense\_count} transactions •

                                  💰 ₹{message.context.total\_expenses?.toFixed(2)}

                                </span>

                              </div>

                            )}

                            <p

                              className={`text-xs mt-2 ${

                                message.type === 'user'

                                  ? 'text-blue-100'

                                  : message.isError

                                  ? 'text-red-500'

                                  : 'text-gray-400'

                              }`}

                            >

                              {formatTimestamp(message.timestamp)}

                            </p>

                          </div>

                        </div>

                      </div>

                    </div>

                  ))

                )}

                {/\* Loading indicator \*/}

                {loading && (

                  <div className="flex justify-start">

                    <div className="bg-white shadow-md rounded-xl p-4 flex items-center gap-3">

                      <div className="w-8 h-8 bg-gradient-to-br from-purple-500 to-pink-500 rounded-full flex items-center justify-center">

                        <MdPsychology className="text-white text-sm" />

                      </div>

                      <div className="flex gap-1">

                        <div className="w-2 h-2 bg-gray-400 rounded-full animate-bounce"></div>

                        <div

                          className="w-2 h-2 bg-gray-400 rounded-full animate-bounce"

                          style={{ animationDelay: '0.1s' }}

                        ></div>

                        <div

                          className="w-2 h-2 bg-gray-400 rounded-full animate-bounce"

                          style={{ animationDelay: '0.2s' }}

                        ></div>

                      </div>

                      <span className="text-sm text-gray-600">AI is thinking...</span>

                    </div>

                  </div>

                )}

                <div ref={conversationEndRef} />

              </div>

              {/\* Input Area \*/}

              <form

                onSubmit={handleQuerySubmit}

                className="p-4 bg-white border-t border-gray-200"

              >

                <div className="flex gap-2">

                  <input

                    type="text"

                    value={query}

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

                    placeholder="Ask about your expenses..."

                    disabled={loading}

                    className="flex-1 px-4 py-3 border border-gray-300 rounded-lg focus:ring-2 focus:ring-purple-500 focus:border-purple-500 disabled:bg-gray-100 disabled:cursor-not-allowed"

                  />

                  <button

                    type="submit"

                    disabled={loading || !query.trim()}

                    className="px-6 py-3 bg-gradient-to-r from-purple-600 to-pink-600 text-white rounded-lg font-medium hover:from-purple-700 hover:to-pink-700 disabled:opacity-50 disabled:cursor-not-allowed transition-all flex items-center gap-2 shadow-lg"

                  >

                    <MdSend className="text-xl" />

                    {loading ? 'Sending...' : 'Send'}

                  </button>

                </div>

              </form>

            </div>

          </div>

          {/\* Sidebar \*/}

          <div className="space-y-6">

            {/\* Quick Questions \*/}

            <div className="bg-white rounded-xl shadow-lg p-6">

              <div className="flex items-center gap-2 mb-4">

                <MdHistory className="text-xl text-purple-600" />

                <h3 className="text-lg font-bold text-gray-900">

                  Quick Questions

                </h3>

              </div>

              <div className="space-y-2">

                {quickQueries.map((quickQuery, index) => (

                  <button

                    key={index}

                    onClick={() => handleQuickQuery(quickQuery)}

                    disabled={loading}

                    className="w-full text-left px-4 py-3 bg-gray-50 hover:bg-purple-50 text-gray-700 hover:text-purple-700 rounded-lg transition-all text-sm border border-transparent hover:border-purple-200 hover:shadow-md disabled:opacity-50 disabled:cursor-not-allowed"

                  >

                    {quickQuery}

                  </button>

                ))}

              </div>

            </div>

            {/\* How It Works \*/}

            <div className="bg-gradient-to-br from-purple-50 to-pink-50 rounded-xl shadow-lg p-6 border border-purple-100">

              <div className="flex items-center gap-2 mb-4">

                <MdTrendingUp className="text-xl text-purple-600" />

                <h3 className="text-lg font-bold text-gray-900">How It Works</h3>

              </div>

              <div className="space-y-3 text-sm text-gray-700">

                <div className="flex gap-3">

                  <div className="flex-shrink-0 w-6 h-6 bg-purple-600 text-white rounded-full flex items-center justify-center text-xs font-bold">

                    1

                  </div>

                  <p>

                    Ask questions in natural language about your expenses

                  </p>

                </div>

                <div className="flex gap-3">

                  <div className="flex-shrink-0 w-6 h-6 bg-purple-600 text-white rounded-full flex items-center justify-center text-xs font-bold">

                    2

                  </div>

                  <p>AI analyzes your spending data based on your query</p>

                </div>

                <div className="flex gap-3">

                  <div className="flex-shrink-0 w-6 h-6 bg-purple-600 text-white rounded-full flex items-center justify-center text-xs font-bold">

                    3

                  </div>

                  <p>Get instant insights with specific numbers and trends</p>

                </div>

              </div>

              <div className="mt-4 p-3 bg-white rounded-lg border border-purple-200">

                <p className="text-xs text-gray-600">

                  <strong className="text-purple-600">💡 Pro Tip:</strong> Include

                  time periods (today, this week, this month) and categories

                  (food, travel) for more accurate results!

                </p>

              </div>

            </div>

            {/\* Examples \*/}

            <div className="bg-white rounded-xl shadow-lg p-6">

              <h3 className="text-lg font-bold text-gray-900 mb-4">

                Example Questions

              </h3>

              <div className="space-y-3 text-sm text-gray-700">

                <div className="p-3 bg-blue-50 rounded-lg border border-blue-200">

                  <p className="font-medium text-blue-900 mb-1">⏰ Time-based</p>

                  <p className="text-blue-700 text-xs">

                    "What did I spend yesterday?"

                  </p>

                </div>

                <div className="p-3 bg-green-50 rounded-lg border border-green-200">

                  <p className="font-medium text-green-900 mb-1">📊 Category-based</p>

                  <p className="text-green-700 text-xs">

                    "Show my food expenses this month"

                  </p>

                </div>

                <div className="p-3 bg-purple-50 rounded-lg border border-purple-200">

                  <p className="font-medium text-purple-900 mb-1">

                    💳 Payment-based

                  </p>

                  <p className="text-purple-700 text-xs">"How much did I pay via GPay?"</p>

                </div>

                <div className="p-3 bg-orange-50 rounded-lg border border-orange-200">

                  <p className="font-medium text-orange-900 mb-1">📈 Analysis</p>

                  <p className="text-orange-700 text-xs">

                    "What's my biggest expense category?"

                  </p>

                </div>

              </div>

            </div>

          </div>

        </div>

      </div>

    </div>

  );

};

export default AiInsights;

E:\expense\_tracker\frontend\src\pages\analytics

E:\expense\_tracker\frontend\src\pages\analytics\Analytics.jsx

/\*\*

 \* Analytics Overview Component

 \* Main analytics page with comprehensive expense insights

 \* Backend: Multiple analytics endpoints

 \*/

import { useState, useEffect } from 'react';

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

import { toast } from 'react-hot-toast';

import {

  MdTrendingUp,

  MdPieChart,

  MdBarChart,

  MdShowChart,

  MdArrowForward,

  MdRefresh,

  MdCalendarToday,

} from 'react-icons/md';

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

import { formatCurrency } from '../../utils/helpers';

const Analytics = () => {

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

  const [refreshing, setRefreshing] = useState(false);

  const [dateFilter, setDateFilter] = useState({

    days: 30,

    label: 'Last 30 Days',

  });

  const [analyticsData, setAnalyticsData] = useState({

    categoryBreakdown: [],

    paymentBreakdown: [],

    crossTab: [],

    dailyTrend: [],

    monthlySummary: null,

  });

  const fetchAnalytics = async (showToast = false) => {

    try {

      if (showToast) {

        setRefreshing(true);

      } else {

        setLoading(true);

      }

      const today = new Date();

      const startDate = new Date(today);

      startDate.setDate(today.getDate() - dateFilter.days);

      const startDateStr = startDate.toISOString().split('T')[0];

      const endDateStr = today.toISOString().split('T')[0];

      console.log('📊 Fetching analytics for:', startDateStr, 'to', endDateStr);

      // Fetch all analytics data in parallel with error handling

      const results = await Promise.allSettled([

        analyticsAPI.getCategoryBreakdown(startDateStr, endDateStr),

        analyticsAPI.getPaymentBreakdown(startDateStr, endDateStr),

        analyticsAPI.getCrossTab(startDateStr, endDateStr),

        analyticsAPI.getDailyTrend(dateFilter.days),

        analyticsAPI.getMonthlySummary(),

      ]);

      // ✅ IMPROVED: Better error handling and data extraction

      const categoryData = results[0].status === 'fulfilled'

        ? (results[0].value.data?.data || [])

        : [];

      const paymentData = results[1].status === 'fulfilled'

        ? (results[1].value.data?.data || [])

        : [];

      const crossTabData = results[2].status === 'fulfilled'

        ? (results[2].value.data?.data || [])

        : [];

      const dailyTrendData = results[3].status === 'fulfilled'

        ? (results[3].value.data?.data || [])

        : [];

      const monthlySummaryData = results[4].status === 'fulfilled'

        ? results[4].value.data

        : null;

      // Log which endpoints failed

      const endpointNames = ['Categories', 'Payments', 'Cross-tab', 'Daily Trend', 'Monthly Summary'];

      results.forEach((result, index) => {

        if (result.status === 'rejected') {

          console.warn(`⚠️ ${endpointNames[index]} failed:`, result.reason?.message);

        } else {

          console.log(`✅ ${endpointNames[index]} loaded successfully`);

        }

      });

      setAnalyticsData({

        categoryBreakdown: categoryData,

        paymentBreakdown: paymentData,

        crossTab: crossTabData,

        dailyTrend: dailyTrendData,

        monthlySummary: monthlySummaryData,

      });

      if (showToast) {

        toast.success('Analytics refreshed');

      }

    } catch (error) {

      console.error('❌ Error fetching analytics:', error);

      toast.error('Failed to load some analytics data');

      // Set empty data so page still renders

      setAnalyticsData({

        categoryBreakdown: [],

        paymentBreakdown: [],

        crossTab: [],

        dailyTrend: [],

        monthlySummary: null,

      });

    } finally {

      setLoading(false);

      setRefreshing(false);

    }

  };

  useEffect(() => {

    fetchAnalytics();

    // eslint-disable-next-line react-hooks/exhaustive-deps

  }, [dateFilter.days]);

  const handleRefresh = () => {

    fetchAnalytics(true);

  };

  const handleDateFilterChange = (days, label) => {

    setDateFilter({ days, label });

  };

  if (loading) {

    return (

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

        <div className="text-center">

          <div className="animate-spin rounded-full h-16 w-16 border-b-2 border-blue-600 mx-auto"></div>

          <p className="mt-4 text-gray-600 font-medium">Loading analytics...</p>

        </div>

      </div>

    );

  }

  // ✅ IMPROVED: Calculate totals with safe fallbacks

  const totalExpenses = analyticsData.categoryBreakdown.reduce(

    (sum, cat) => sum + (parseFloat(cat.total) || 0),

    0

  );

  const totalTransactions = analyticsData.categoryBreakdown.reduce(

    (sum, cat) => sum + (parseInt(cat.count) || 0),

    0

  );

  const averagePerTransaction = totalTransactions > 0

    ? totalExpenses / totalTransactions

    : 0;

  // ✅ ADDED: Get current month data

  const getCurrentMonthData = () => {

    if (!analyticsData.monthlySummary?.data) return null;

    const currentMonth = new Date().getMonth() + 1;

    const monthData = analyticsData.monthlySummary.data.find(

      m => m.month\_number === currentMonth

    );

    return monthData || null;

  };

  const currentMonthData = getCurrentMonthData();

  return (

    <div className="min-h-screen bg-gray-50 p-6">

      <div className="max-w-7xl mx-auto">

        {/\* Header \*/}

        <div className="flex flex-col md:flex-row md:items-center md:justify-between mb-8">

          <div>

            <h1 className="text-3xl font-bold text-gray-900">Analytics</h1>

            <p className="mt-1 text-sm text-gray-600">

              Detailed insights into your spending patterns - {dateFilter.label}

            </p>

          </div>

          <div className="mt-4 md:mt-0 flex flex-wrap gap-3">

            {/\* Date Filter Buttons \*/}

            <div className="flex gap-2 bg-white rounded-lg p-1 shadow-sm">

              <button

                onClick={() => handleDateFilterChange(7, 'Last 7 Days')}

                className={`px-4 py-2 rounded-md text-sm font-medium transition-colors ${

                  dateFilter.days === 7

                    ? 'bg-blue-600 text-white'

                    : 'text-gray-600 hover:bg-gray-100'

                }`}

              >

                7D

              </button>

              <button

                onClick={() => handleDateFilterChange(30, 'Last 30 Days')}

                className={`px-4 py-2 rounded-md text-sm font-medium transition-colors ${

                  dateFilter.days === 30

                    ? 'bg-blue-600 text-white'

                    : 'text-gray-600 hover:bg-gray-100'

                }`}

              >

                30D

              </button>

              <button

                onClick={() => handleDateFilterChange(90, 'Last 90 Days')}

                className={`px-4 py-2 rounded-md text-sm font-medium transition-colors ${

                  dateFilter.days === 90

                    ? 'bg-blue-600 text-white'

                    : 'text-gray-600 hover:bg-gray-100'

                }`}

              >

                90D

              </button>

              <button

                onClick={() => handleDateFilterChange(365, 'Last Year')}

                className={`px-4 py-2 rounded-md text-sm font-medium transition-colors ${

                  dateFilter.days === 365

                    ? 'bg-blue-600 text-white'

                    : 'text-gray-600 hover:bg-gray-100'

                }`}

              >

                1Y

              </button>

            </div>

            {/\* Refresh Button \*/}

            <button

              onClick={handleRefresh}

              disabled={refreshing}

              className="flex items-center gap-2 px-4 py-2 bg-white rounded-lg shadow-sm text-gray-700 hover:bg-gray-50 transition-colors disabled:opacity-50 disabled:cursor-not-allowed"

            >

              <MdRefresh

                className={`text-xl ${refreshing ? 'animate-spin' : ''}`}

              />

              Refresh

            </button>

          </div>

        </div>

        {/\* Summary Cards \*/}

        <div className="grid grid-cols-1 md:grid-cols-3 gap-6 mb-8">

          <div className="bg-gradient-to-br from-blue-500 to-blue-600 rounded-xl shadow-lg p-6 text-white">

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

              <div>

                <p className="text-blue-100 text-sm font-medium">Total Spent</p>

                <h3 className="text-3xl font-bold mt-2">

                  {formatCurrency(totalExpenses)}

                </h3>

              </div>

              <div className="bg-blue-400 bg-opacity-50 rounded-full p-3">

                <MdTrendingUp className="text-3xl" />

              </div>

            </div>

            <p className="text-blue-100 text-xs mt-3">{dateFilter.label}</p>

          </div>

          <div className="bg-gradient-to-br from-purple-500 to-purple-600 rounded-xl shadow-lg p-6 text-white">

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

              <div>

                <p className="text-purple-100 text-sm font-medium">

                  Transactions

                </p>

                <h3 className="text-3xl font-bold mt-2">{totalTransactions}</h3>

              </div>

              <div className="bg-purple-400 bg-opacity-50 rounded-full p-3">

                <MdBarChart className="text-3xl" />

              </div>

            </div>

            <p className="text-purple-100 text-xs mt-3">Total count</p>

          </div>

          <div className="bg-gradient-to-br from-green-500 to-green-600 rounded-xl shadow-lg p-6 text-white">

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

              <div>

                <p className="text-green-100 text-sm font-medium">Avg/Transaction</p>

                <h3 className="text-3xl font-bold mt-2">

                  {formatCurrency(averagePerTransaction)}

                </h3>

              </div>

              <div className="bg-green-400 bg-opacity-50 rounded-full p-3">

                <MdShowChart className="text-3xl" />

              </div>

            </div>

            <p className="text-green-100 text-xs mt-3">Per expense</p>

          </div>

        </div>

        {/\* ✅ IMPROVED: Monthly Summary with better data handling \*/}

        {currentMonthData && (

          <div className="bg-white rounded-xl shadow-lg p-6 mb-8">

            <div className="flex items-center justify-between mb-6">

              <h2 className="text-xl font-bold text-gray-900 flex items-center gap-2">

                <MdCalendarToday className="text-blue-600" />

                This Month's Summary

              </h2>

              <span className="text-sm text-gray-600">{currentMonthData.month}</span>

            </div>

            <div className="grid grid-cols-1 md:grid-cols-3 gap-6">

              <div className="text-center p-4 bg-blue-50 rounded-lg">

                <p className="text-sm text-gray-600 mb-2">Total Expenses</p>

                <p className="text-2xl font-bold text-blue-600">

                  {formatCurrency(currentMonthData.total || 0)}

                </p>

              </div>

              <div className="text-center p-4 bg-purple-50 rounded-lg">

                <p className="text-sm text-gray-600 mb-2">Transactions</p>

                <p className="text-2xl font-bold text-purple-600">

                  {currentMonthData.count || 0}

                </p>

              </div>

              <div className="text-center p-4 bg-green-50 rounded-lg">

                <p className="text-sm text-gray-600 mb-2">Average</p>

                <p className="text-2xl font-bold text-green-600">

                  {formatCurrency(currentMonthData.average || 0)}

                </p>

              </div>

            </div>

          </div>

        )}

        {/\* Quick Links \*/}

        <div className="grid grid-cols-1 md:grid-cols-3 gap-6 mb-8">

          {/\* Category Breakdown Link \*/}

          <Link

            to="/analytics/category"

            className="bg-white rounded-xl shadow-lg p-6 hover:shadow-xl transition-shadow group"

          >

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

              <div className="w-12 h-12 bg-blue-100 rounded-lg flex items-center justify-center">

                <MdPieChart className="text-2xl text-blue-600" />

              </div>

              <MdArrowForward className="text-gray-400 group-hover:text-blue-600 transition-colors" />

            </div>

            <h3 className="text-lg font-bold text-gray-900 mb-2">

              Category Breakdown

            </h3>

            <p className="text-sm text-gray-600">

              View detailed spending by category with charts and percentages

            </p>

            {analyticsData.categoryBreakdown.length > 0 && (

              <p className="text-xs text-blue-600 font-medium mt-3">

                {analyticsData.categoryBreakdown.length} categories tracked

              </p>

            )}

          </Link>

          {/\* Payment Analysis Link \*/}

          <Link

            to="/analytics/payment"

            className="bg-white rounded-xl shadow-lg p-6 hover:shadow-xl transition-shadow group"

          >

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

              <div className="w-12 h-12 bg-purple-100 rounded-lg flex items-center justify-center">

                <MdBarChart className="text-2xl text-purple-600" />

              </div>

              <MdArrowForward className="text-gray-400 group-hover:text-purple-600 transition-colors" />

            </div>

            <h3 className="text-lg font-bold text-gray-900 mb-2">

              Payment Analysis

            </h3>

            <p className="text-sm text-gray-600">

              Analyze spending patterns across different payment methods

            </p>

            {analyticsData.paymentBreakdown.length > 0 && (

              <p className="text-xs text-purple-600 font-medium mt-3">

                {analyticsData.paymentBreakdown.length} payment methods used

              </p>

            )}

          </Link>

          {/\* Trend Analysis Link \*/}

          <Link

            to="/analytics/trends"

            className="bg-white rounded-xl shadow-lg p-6 hover:shadow-xl transition-shadow group"

          >

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

              <div className="w-12 h-12 bg-green-100 rounded-lg flex items-center justify-center">

                <MdShowChart className="text-2xl text-green-600" />

              </div>

              <MdArrowForward className="text-gray-400 group-hover:text-green-600 transition-colors" />

            </div>

            <h3 className="text-lg font-bold text-gray-900 mb-2">

              Trend Analysis

            </h3>

            <p className="text-sm text-gray-600">

              Track spending trends over time with daily and monthly views

            </p>

            {analyticsData.dailyTrend.length > 0 && (

              <p className="text-xs text-green-600 font-medium mt-3">

                {analyticsData.dailyTrend.length} days of data

              </p>

            )}

          </Link>

        </div>

        {/\* Preview Sections \*/}

        <div className="grid grid-cols-1 lg:grid-cols-2 gap-6">

          {/\* Top Categories Preview \*/}

          <div className="bg-white rounded-xl shadow-lg p-6">

            <div className="flex items-center justify-between mb-6">

              <h2 className="text-xl font-bold text-gray-900">Top Categories</h2>

              <Link

                to="/analytics/category"

                className="text-blue-600 hover:text-blue-700 text-sm font-medium flex items-center gap-1"

              >

                View All <MdArrowForward />

              </Link>

            </div>

            {analyticsData.categoryBreakdown.length > 0 ? (

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

                {analyticsData.categoryBreakdown.slice(0, 5).map((category, index) => (

                  <div key={index} className="space-y-2">

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

                      <div className="flex items-center gap-3 flex-1 min-w-0">

                        <div

                          className="w-3 h-3 rounded-full flex-shrink-0"

                          style={{ backgroundColor: category.color || '#3B82F6' }}

                        ></div>

                        <span className="text-sm font-medium text-gray-900 truncate">

                          {category.category}

                        </span>

                        <span className="text-xs text-gray-500">

                          ({category.count || 0})

                        </span>

                      </div>

                      <div className="text-right ml-2">

                        <p className="text-sm font-bold text-gray-900 whitespace-nowrap">

                          {formatCurrency(category.total || 0)}

                        </p>

                        <p className="text-xs text-gray-500">

                          {(category.percentage || 0).toFixed(1)}%

                        </p>

                      </div>

                    </div>

                    <div className="w-full bg-gray-200 rounded-full h-2">

                      <div

                        className="h-2 rounded-full transition-all duration-300"

                        style={{

                          width: `${category.percentage || 0}%`,

                          backgroundColor: category.color || '#3B82F6',

                        }}

                      ></div>

                    </div>

                  </div>

                ))}

              </div>

            ) : (

              <div className="text-center py-8 text-gray-400">

                <MdPieChart className="text-5xl mx-auto mb-2 opacity-50" />

                <p>No category data available</p>

                <Link

                  to="/expenses/add"

                  className="text-blue-600 hover:text-blue-700 text-sm font-medium mt-2 inline-block"

                >

                  Add your first expense

                </Link>

              </div>

            )}

          </div>

          {/\* Top Payment Methods Preview \*/}

          <div className="bg-white rounded-xl shadow-lg p-6">

            <div className="flex items-center justify-between mb-6">

              <h2 className="text-xl font-bold text-gray-900">Top Payment Methods</h2>

              <Link

                to="/analytics/payment"

                className="text-blue-600 hover:text-blue-700 text-sm font-medium flex items-center gap-1"

              >

                View All <MdArrowForward />

              </Link>

            </div>

            {analyticsData.paymentBreakdown.length > 0 ? (

              <div className="space-y-3">

                {analyticsData.paymentBreakdown.slice(0, 5).map((payment, index) => (

                  <div key={index} className="flex items-center justify-between p-3 bg-gray-50 rounded-lg hover:bg-gray-100 transition-colors">

                    <div className="flex-1 min-w-0">

                      <p className="text-sm font-medium text-gray-900 truncate">

                        {payment.paymentmode || payment.payment\_mode}

                      </p>

                      <p className="text-xs text-gray-500">

                        {payment.count || 0} transactions

                      </p>

                    </div>

                    <div className="text-right ml-2">

                      <p className="text-sm font-bold text-gray-900 whitespace-nowrap">

                        {formatCurrency(payment.total || 0)}

                      </p>

                      <p className="text-xs text-gray-500">

                        {(payment.percentage || 0).toFixed(1)}%

                      </p>

                    </div>

                  </div>

                ))}

              </div>

            ) : (

              <div className="text-center py-8 text-gray-400">

                <MdBarChart className="text-5xl mx-auto mb-2 opacity-50" />

                <p>No payment data available</p>

              </div>

            )}

          </div>

        </div>

      </div>

    </div>

  );

};

export default Analytics;

E:\expense\_tracker\frontend\src\pages\analytics\CategoryBreakdown.jsx

/\*\*

 \* CategoryBreakdown Component

 \* Detailed category-wise expense analysis with pie chart

 \* Backend: GET /api/analytics/categories-vs-expenses

 \*/

import { useState, useEffect } from 'react';

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

import { toast } from 'react-hot-toast';

import { MdArrowBack, MdRefresh, MdPieChart } from 'react-icons/md';

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

import { formatCurrency } from '../../utils/helpers';

const CategoryBreakdown = () => {

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

  const [refreshing, setRefreshing] = useState(false);

  const [categories, setCategories] = useState([]);

  // ✅ FIXED: Initialize with proper default dates

  const getDefaultDates = () => {

    const today = new Date();

    const thirtyDaysAgo = new Date();

    thirtyDaysAgo.setDate(today.getDate() - 30);

    return {

      startDate: thirtyDaysAgo.toISOString().split('T')[0],

      endDate: today.toISOString().split('T')[0],

    };

  };

  const [dateRange, setDateRange] = useState(getDefaultDates());

  const fetchCategoryBreakdown = async (showToast = false) => {

    try {

      if (showToast) {

        setRefreshing(true);

      } else {

        setLoading(true);

      }

      console.log('📊 Fetching category breakdown:', dateRange);

      // ✅ FIXED: Ensure dates are valid

      const startDate = dateRange.startDate || getDefaultDates().startDate;

      const endDate = dateRange.endDate || getDefaultDates().endDate;

      const response = await analyticsAPI.getCategoryBreakdown(startDate, endDate);

      console.log('✅ Category breakdown response:', response.data);

      // ✅ FIXED: Proper data extraction with multiple fallbacks

      const categoryData = response.data?.data || response.data || [];

      console.log('📋 Extracted category data:', categoryData);

      if (Array.isArray(categoryData) && categoryData.length > 0) {

        setCategories(categoryData);

        if (showToast) {

          toast.success('Data refreshed successfully');

        }

      } else {

        setCategories([]);

        if (showToast) {

          toast.info('No category data found for selected period');

        }

      }

    } catch (error) {

      console.error('❌ Error fetching category breakdown:', error);

      console.error('Error details:', {

        message: error.message,

        response: error.response?.data,

        status: error.response?.status,

      });

      toast.error(

        error.response?.data?.error ||

        'Failed to load category breakdown'

      );

      setCategories([]);

    } finally {

      setLoading(false);

      setRefreshing(false);

    }

  };

  // ✅ FIXED: Only fetch on mount, not on every dateRange change

  useEffect(() => {

    fetchCategoryBreakdown();

    // eslint-disable-next-line react-hooks/exhaustive-deps

  }, []); // Empty dependency array - only run once on mount

  // ✅ ADDED: Handler for date change with manual trigger

  const handleDateChange = (field, value) => {

    setDateRange(prev => ({

      ...prev,

      [field]: value

    }));

  };

  // ✅ ADDED: Handler for applying date filter

  const handleApplyFilter = () => {

    fetchCategoryBreakdown(true);

  };

  // ✅ FIXED: Safe calculations with proper number conversion

  const totalAmount = categories.reduce(

    (sum, cat) => sum + (parseFloat(cat.total) || 0),

    0

  );

  const totalCount = categories.reduce(

    (sum, cat) => sum + (parseInt(cat.count) || 0),

    0

  );

  // Generate colors for pie chart

  const chartColors = [

    '#3B82F6', '#8B5CF6', '#EC4899', '#F59E0B', '#10B981',

    '#6366F1', '#F97316', '#14B8A6', '#EF4444', '#84CC16',

  ];

  if (loading) {

    return (

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

        <div className="text-center">

          <div className="animate-spin rounded-full h-16 w-16 border-b-2 border-blue-600 mx-auto"></div>

          <p className="mt-4 text-gray-600 font-medium">Loading category breakdown...</p>

        </div>

      </div>

    );

  }

  return (

    <div className="min-h-screen bg-gray-50 p-6">

      <div className="max-w-7xl mx-auto">

        {/\* Header \*/}

        <div className="mb-6">

          <Link

            to="/analytics"

            className="inline-flex items-center gap-2 text-blue-600 hover:text-blue-700 mb-4 font-medium"

          >

            <MdArrowBack className="text-xl" />

            Back to Analytics

          </Link>

          <div className="flex flex-col md:flex-row md:items-center md:justify-between">

            <div>

              <h1 className="text-3xl font-bold text-gray-900">

                Category Breakdown

              </h1>

              <p className="mt-1 text-sm text-gray-600">

                Detailed analysis of expenses by category

              </p>

            </div>

            <button

              onClick={() => fetchCategoryBreakdown(true)}

              disabled={refreshing}

              className="mt-4 md:mt-0 flex items-center gap-2 px-4 py-2 bg-white rounded-lg shadow-sm text-gray-700 hover:bg-gray-50 transition-colors disabled:opacity-50 disabled:cursor-not-allowed"

            >

              <MdRefresh className={`text-xl ${refreshing ? 'animate-spin' : ''}`} />

              {refreshing ? 'Refreshing...' : 'Refresh'}

            </button>

          </div>

        </div>

        {/\* Date Range Filter \*/}

        <div className="bg-white rounded-xl shadow-lg p-6 mb-6">

          <h3 className="text-lg font-bold text-gray-900 mb-4">Date Range</h3>

          <div className="grid grid-cols-1 md:grid-cols-3 gap-4">

            <div>

              <label className="block text-sm font-medium text-gray-700 mb-2">

                Start Date

              </label>

              <input

                type="date"

                value={dateRange.startDate}

                onChange={(e) => handleDateChange('startDate', e.target.value)}

                max={dateRange.endDate}

                className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500"

              />

            </div>

            <div>

              <label className="block text-sm font-medium text-gray-700 mb-2">

                End Date

              </label>

              <input

                type="date"

                value={dateRange.endDate}

                onChange={(e) => handleDateChange('endDate', e.target.value)}

                min={dateRange.startDate}

                max={new Date().toISOString().split('T')[0]}

                className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500"

              />

            </div>

            <div className="flex items-end">

              <button

                onClick={handleApplyFilter}

                disabled={refreshing}

                className="w-full px-4 py-2 bg-blue-600 text-white rounded-lg font-medium hover:bg-blue-700 transition-colors disabled:opacity-50 disabled:cursor-not-allowed"

              >

                {refreshing ? 'Loading...' : 'Apply Filter'}

              </button>

            </div>

          </div>

        </div>

        {/\* Summary Cards \*/}

        <div className="grid grid-cols-1 md:grid-cols-3 gap-6 mb-6">

          <div className="bg-white rounded-xl shadow-lg p-6">

            <p className="text-sm text-gray-600 mb-2">Total Amount</p>

            <p className="text-3xl font-bold text-blue-600">

              {formatCurrency(totalAmount)}

            </p>

            <p className="text-xs text-gray-500 mt-2">

              From {categories.length} categories

            </p>

          </div>

          <div className="bg-white rounded-xl shadow-lg p-6">

            <p className="text-sm text-gray-600 mb-2">Total Transactions</p>

            <p className="text-3xl font-bold text-purple-600">{totalCount}</p>

            <p className="text-xs text-gray-500 mt-2">

              Across all categories

            </p>

          </div>

          <div className="bg-white rounded-xl shadow-lg p-6">

            <p className="text-sm text-gray-600 mb-2">Average per Transaction</p>

            <p className="text-3xl font-bold text-green-600">

              {formatCurrency(totalCount > 0 ? totalAmount / totalCount : 0)}

            </p>

            <p className="text-xs text-gray-500 mt-2">

              Overall average

            </p>

          </div>

        </div>

        {/\* Category Details \*/}

        {categories.length > 0 ? (

          <div className="grid grid-cols-1 lg:grid-cols-2 gap-6">

            {/\* Pie Chart Visualization \*/}

            <div className="bg-white rounded-xl shadow-lg p-6">

              <h3 className="text-xl font-bold text-gray-900 mb-6">

                Visual Distribution

              </h3>

              <div className="relative w-full aspect-square max-w-sm mx-auto">

                {/\* Simple Donut Chart \*/}

                <svg viewBox="0 0 200 200" className="w-full h-full">

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

                    const percentage = parseFloat(category.percentage) || 0;

                    const startAngle = categories

                      .slice(0, index)

                      .reduce((sum, cat) => sum + ((parseFloat(cat.percentage) || 0) / 100) \* 360, 0);

                    const angle = (percentage / 100) \* 360;

                    const endAngle = startAngle + angle;

                    const startRad = (startAngle - 90) \* (Math.PI / 180);

                    const endRad = (endAngle - 90) \* (Math.PI / 180);

                    const x1 = 100 + 80 \* Math.cos(startRad);

                    const y1 = 100 + 80 \* Math.sin(startRad);

                    const x2 = 100 + 80 \* Math.cos(endRad);

                    const y2 = 100 + 80 \* Math.sin(endRad);

                    const largeArc = angle > 180 ? 1 : 0;

                    return (

                      <path

                        key={index}

                        d={`M 100 100 L ${x1} ${y1} A 80 80 0 ${largeArc} 1 ${x2} ${y2} Z`}

                        fill={category.color || chartColors[index % chartColors.length]}

                        className="hover:opacity-80 transition-opacity cursor-pointer"

                        title={`${category.category}: ${percentage.toFixed(1)}%`}

                      />

                    );

                  })}

                  {/\* Center circle for donut effect \*/}

                  <circle cx="100" cy="100" r="50" fill="white" />

                  <text

                    x="100"

                    y="95"

                    textAnchor="middle"

                    className="text-xs fill-gray-600 font-medium"

                  >

                    Total

                  </text>

                  <text

                    x="100"

                    y="110"

                    textAnchor="middle"

                    className="text-sm fill-gray-900 font-bold"

                  >

                    {categories.length}

                  </text>

                </svg>

              </div>

              {/\* Legend \*/}

              <div className="mt-6 grid grid-cols-2 gap-2">

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

                  <div key={index} className="flex items-center gap-2">

                    <div

                      className="w-3 h-3 rounded-full flex-shrink-0"

                      style={{

                        backgroundColor:

                          category.color || chartColors[index % chartColors.length],

                      }}

                    ></div>

                    <span className="text-xs text-gray-700 truncate">

                      {category.category}

                    </span>

                  </div>

                ))}

              </div>

            </div>

            {/\* Category List \*/}

            <div className="bg-white rounded-xl shadow-lg p-6">

              <h3 className="text-xl font-bold text-gray-900 mb-6">

                Category Details

              </h3>

              <div className="space-y-4 max-h-[500px] overflow-y-auto pr-2">

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

                  const categoryTotal = parseFloat(category.total) || 0;

                  const categoryCount = parseInt(category.count) || 0;

                  const categoryPercentage = parseFloat(category.percentage) || 0;

                  const avgPerTransaction = categoryCount > 0 ? categoryTotal / categoryCount : 0;

                  return (

                    <div key={index} className="border-b border-gray-200 pb-4 last:border-0">

                      <div className="flex items-start justify-between mb-2">

                        <div className="flex items-center gap-3 flex-1 min-w-0">

                          <div

                            className="w-4 h-4 rounded-full flex-shrink-0 mt-1"

                            style={{

                              backgroundColor:

                                category.color || chartColors[index % chartColors.length],

                            }}

                          ></div>

                          <div className="flex-1 min-w-0">

                            <p className="font-medium text-gray-900 truncate">

                              {category.category}

                            </p>

                            <p className="text-xs text-gray-500 mt-1">

                              {categoryCount} transactions

                            </p>

                          </div>

                        </div>

                        <div className="text-right ml-2">

                          <p className="font-bold text-gray-900 whitespace-nowrap">

                            {formatCurrency(categoryTotal)}

                          </p>

                          <p className="text-sm text-blue-600 font-medium">

                            {categoryPercentage.toFixed(1)}%

                          </p>

                        </div>

                      </div>

                      {/\* Progress Bar \*/}

                      <div className="w-full bg-gray-200 rounded-full h-2 mt-2">

                        <div

                          className="h-2 rounded-full transition-all duration-300"

                          style={{

                            width: `${categoryPercentage}%`,

                            backgroundColor:

                              category.color || chartColors[index % chartColors.length],

                          }}

                        ></div>

                      </div>

                      {/\* Average per transaction \*/}

                      <p className="text-xs text-gray-500 mt-2">

                        Avg: {formatCurrency(avgPerTransaction)} per transaction

                      </p>

                    </div>

                  );

                })}

              </div>

            </div>

          </div>

        ) : (

          <div className="bg-white rounded-xl shadow-lg p-12 text-center">

            <MdPieChart className="mx-auto text-6xl text-gray-300 mb-4" />

            <p className="text-gray-500 text-lg font-medium mb-2">No category data available</p>

            <p className="text-gray-400 text-sm mb-4">

              Try adjusting the date range or add some expenses

            </p>

            <Link

              to="/expenses/add"

              className="inline-block px-6 py-3 bg-blue-600 text-white rounded-lg font-medium hover:bg-blue-700 transition-colors"

            >

              Add Expense

            </Link>

          </div>

        )}

      </div>

    </div>

  );

};

export default CategoryBreakdown;

E:\expense\_tracker\frontend\src\pages\analytics\PaymentAnalysis.jsx

/\*\*

 \* PaymentAnalysis Component

 \* Detailed payment mode analysis with cross-tabulation

 \* Backend: GET /api/analytics/payment-modes-vs-expenses, /api/analytics/payment-modes-vs-categories

 \*/

import { useState, useEffect } from 'react';

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

import { toast } from 'react-hot-toast';

import { MdArrowBack, MdRefresh, MdPayment } from 'react-icons/md';

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

import { formatCurrency } from '../../utils/helpers';

const PaymentAnalysis = () => {

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

  const [refreshing, setRefreshing] = useState(false);

  const [paymentModes, setPaymentModes] = useState([]);

  const [crossTabData, setCrossTabData] = useState([]);

  // ✅ FIXED: Initialize with proper default dates

  const getDefaultDates = () => {

    const today = new Date();

    const thirtyDaysAgo = new Date();

    thirtyDaysAgo.setDate(today.getDate() - 30);

    return {

      startDate: thirtyDaysAgo.toISOString().split('T')[0],

      endDate: today.toISOString().split('T')[0],

    };

  };

  const [dateRange, setDateRange] = useState(getDefaultDates());

  const fetchPaymentAnalysis = async (showToast = false) => {

    try {

      if (showToast) {

        setRefreshing(true);

      } else {

        setLoading(true);

      }

      console.log('📊 Fetching payment analysis:', dateRange);

      // ✅ FIXED: Ensure dates are valid

      const startDate = dateRange.startDate || getDefaultDates().startDate;

      const endDate = dateRange.endDate || getDefaultDates().endDate;

      // ✅ FIXED: Use Promise.allSettled for partial failure handling

      const results = await Promise.allSettled([

        analyticsAPI.getPaymentBreakdown(startDate, endDate),

        analyticsAPI.getCrossTab(startDate, endDate),

      ]);

      console.log('✅ Payment analysis results:', results);

      // ✅ FIXED: Safe data extraction with multiple fallbacks

      const paymentData = results[0].status === 'fulfilled'

        ? (results[0].value.data?.data || results[0].value.data || [])

        : [];

      const crossData = results[1].status === 'fulfilled'

        ? (results[1].value.data?.data || results[1].value.data || [])

        : [];

      console.log('📋 Extracted payment data:', paymentData);

      console.log('📋 Extracted cross-tab data:', crossData);

      if (Array.isArray(paymentData)) {

        setPaymentModes(paymentData);

      } else {

        console.warn('⚠️ Payment data is not an array:', paymentData);

        setPaymentModes([]);

      }

      if (Array.isArray(crossData)) {

        setCrossTabData(crossData);

      } else {

        console.warn('⚠️ Cross-tab data is not an array:', crossData);

        setCrossTabData([]);

      }

      if (showToast) {

        if (paymentData.length > 0 || crossData.length > 0) {

          toast.success('Data refreshed successfully');

        } else {

          toast.info('No payment data found for selected period');

        }

      }

    } catch (error) {

      console.error('❌ Error fetching payment analysis:', error);

      console.error('Error details:', {

        message: error.message,

        response: error.response?.data,

        status: error.response?.status,

      });

      toast.error(

        error.response?.data?.error ||

        'Failed to load payment analysis'

      );

      setPaymentModes([]);

      setCrossTabData([]);

    } finally {

      setLoading(false);

      setRefreshing(false);

    }

  };

  // ✅ FIXED: Only fetch on mount, not on every dateRange change

  useEffect(() => {

    fetchPaymentAnalysis();

    // eslint-disable-next-line react-hooks/exhaustive-deps

  }, []); // Empty dependency array - only run once on mount

  // ✅ ADDED: Handler for date change with manual trigger

  const handleDateChange = (field, value) => {

    setDateRange(prev => ({

      ...prev,

      [field]: value

    }));

  };

  // ✅ ADDED: Handler for applying date filter

  const handleApplyFilter = () => {

    fetchPaymentAnalysis(true);

  };

  // ✅ FIXED: Safe calculations with proper number conversion

  const totalAmount = paymentModes.reduce(

    (sum, pm) => sum + (parseFloat(pm.total) || 0),

    0

  );

  const totalCount = paymentModes.reduce(

    (sum, pm) => sum + (parseInt(pm.count) || 0),

    0

  );

  if (loading) {

    return (

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

        <div className="text-center">

          <div className="animate-spin rounded-full h-16 w-16 border-b-2 border-blue-600 mx-auto"></div>

          <p className="mt-4 text-gray-600 font-medium">Loading payment analysis...</p>

        </div>

      </div>

    );

  }

  return (

    <div className="min-h-screen bg-gray-50 p-6">

      <div className="max-w-7xl mx-auto">

        {/\* Header \*/}

        <div className="mb-6">

          <Link

            to="/analytics"

            className="inline-flex items-center gap-2 text-blue-600 hover:text-blue-700 mb-4 font-medium"

          >

            <MdArrowBack className="text-xl" />

            Back to Analytics

          </Link>

          <div className="flex flex-col md:flex-row md:items-center md:justify-between">

            <div>

              <h1 className="text-3xl font-bold text-gray-900">

                Payment Analysis

              </h1>

              <p className="mt-1 text-sm text-gray-600">

                Analyze spending across different payment methods

              </p>

            </div>

            <button

              onClick={() => fetchPaymentAnalysis(true)}

              disabled={refreshing}

              className="mt-4 md:mt-0 flex items-center gap-2 px-4 py-2 bg-white rounded-lg shadow-sm text-gray-700 hover:bg-gray-50 transition-colors disabled:opacity-50 disabled:cursor-not-allowed"

            >

              <MdRefresh className={`text-xl ${refreshing ? 'animate-spin' : ''}`} />

              {refreshing ? 'Refreshing...' : 'Refresh'}

            </button>

          </div>

        </div>

        {/\* Date Range Filter \*/}

        <div className="bg-white rounded-xl shadow-lg p-6 mb-6">

          <h3 className="text-lg font-bold text-gray-900 mb-4">Date Range</h3>

          <div className="grid grid-cols-1 md:grid-cols-3 gap-4">

            <div>

              <label className="block text-sm font-medium text-gray-700 mb-2">

                Start Date

              </label>

              <input

                type="date"

                value={dateRange.startDate}

                onChange={(e) => handleDateChange('startDate', e.target.value)}

                max={dateRange.endDate}

                className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500"

              />

            </div>

            <div>

              <label className="block text-sm font-medium text-gray-700 mb-2">

                End Date

              </label>

              <input

                type="date"

                value={dateRange.endDate}

                onChange={(e) => handleDateChange('endDate', e.target.value)}

                min={dateRange.startDate}

                max={new Date().toISOString().split('T')[0]}

                className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500"

              />

            </div>

            <div className="flex items-end">

              <button

                onClick={handleApplyFilter}

                disabled={refreshing}

                className="w-full px-4 py-2 bg-blue-600 text-white rounded-lg font-medium hover:bg-blue-700 transition-colors disabled:opacity-50 disabled:cursor-not-allowed"

              >

                {refreshing ? 'Loading...' : 'Apply Filter'}

              </button>

            </div>

          </div>

        </div>

        {/\* Summary Cards \*/}

        <div className="grid grid-cols-1 md:grid-cols-3 gap-6 mb-6">

          <div className="bg-white rounded-xl shadow-lg p-6">

            <p className="text-sm text-gray-600 mb-2">Total Amount</p>

            <p className="text-3xl font-bold text-blue-600">

              {formatCurrency(totalAmount)}

            </p>

            <p className="text-xs text-gray-500 mt-2">

              From {paymentModes.length} payment methods

            </p>

          </div>

          <div className="bg-white rounded-xl shadow-lg p-6">

            <p className="text-sm text-gray-600 mb-2">Total Transactions</p>

            <p className="text-3xl font-bold text-purple-600">{totalCount}</p>

            <p className="text-xs text-gray-500 mt-2">

              Across all payment methods

            </p>

          </div>

          <div className="bg-white rounded-xl shadow-lg p-6">

            <p className="text-sm text-gray-600 mb-2">Average per Transaction</p>

            <p className="text-3xl font-bold text-green-600">

              {formatCurrency(totalCount > 0 ? totalAmount / totalCount : 0)}

            </p>

            <p className="text-xs text-gray-500 mt-2">

              Overall average

            </p>

          </div>

        </div>

        {/\* Payment Mode Breakdown \*/}

        {paymentModes.length > 0 ? (

          <>

            <div className="bg-white rounded-xl shadow-lg p-6 mb-6">

              <h3 className="text-xl font-bold text-gray-900 mb-6">

                Payment Mode Breakdown

              </h3>

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

                {paymentModes.map((payment, index) => {

                  // ✅ FIXED: Safe values with proper type conversion

                  const total = parseFloat(payment.total) || 0;

                  const count = parseInt(payment.count) || 0;

                  const percentage = parseFloat(payment.percentage) || 0;

                  const avgPerTransaction = count > 0 ? total / count : 0;

                  return (

                    <div key={index} className="border-b border-gray-200 pb-4 last:border-0">

                      <div className="flex items-center justify-between mb-2">

                        <div className="flex-1 min-w-0">

                          <p className="font-medium text-gray-900 truncate">

                            {payment.paymentmode || payment.payment\_mode || 'Unknown'}

                          </p>

                          <p className="text-xs text-gray-500 mt-1">

                            {count} transactions

                          </p>

                        </div>

                        <div className="text-right ml-2">

                          <p className="font-bold text-gray-900 whitespace-nowrap">

                            {formatCurrency(total)}

                          </p>

                          <p className="text-sm text-purple-600 font-medium">

                            {percentage.toFixed(1)}%

                          </p>

                        </div>

                      </div>

                      {/\* Progress Bar \*/}

                      <div className="w-full bg-gray-200 rounded-full h-3 mt-2">

                        <div

                          className="bg-gradient-to-r from-purple-500 to-pink-500 h-3 rounded-full transition-all duration-300"

                          style={{ width: `${Math.min(percentage, 100)}%` }}

                        ></div>

                      </div>

                      {/\* Average per transaction \*/}

                      <div className="flex justify-between mt-2 text-xs text-gray-500">

                        <span>Avg: {formatCurrency(avgPerTransaction)}</span>

                        <span>{count} uses</span>

                      </div>

                    </div>

                  );

                })}

              </div>

            </div>

            {/\* Cross-Tabulation: Payment Modes vs Categories \*/}

            {crossTabData.length > 0 && (

              <div className="bg-white rounded-xl shadow-lg p-6">

                <h3 className="text-xl font-bold text-gray-900 mb-6">

                  Payment Modes vs Categories

                </h3>

                <p className="text-sm text-gray-600 mb-6">

                  See how different payment methods are used across spending categories

                </p>

                <div className="space-y-6">

                  {crossTabData.map((categoryData, index) => (

                    <div key={index} className="border-b border-gray-200 pb-6 last:border-0">

                      <h4 className="font-bold text-gray-900 mb-4 text-lg">

                        {categoryData.category || 'Unknown Category'}

                      </h4>

                      {/\* ✅ FIXED: Safe access to payments object with better handling \*/}

                      {categoryData.payments && typeof categoryData.payments === 'object' ? (

                        <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-4">

                          {Object.entries(categoryData.payments).map(

                            ([paymentMode, paymentData], pmIndex) => {

                              // ✅ FIXED: Handle both old format (number) and new format (object)

                              const total = typeof paymentData === 'object'

                                ? parseFloat(paymentData.total) || 0

                                : parseFloat(paymentData) || 0;

                              const count = typeof paymentData === 'object'

                                ? parseInt(paymentData.count) || 0

                                : 0;

                              return (

                                <div

                                  key={pmIndex}

                                  className="bg-gradient-to-br from-purple-50 to-pink-50 rounded-lg p-4 hover:shadow-md transition-all border border-purple-100"

                                >

                                  <p className="text-sm text-gray-600 mb-1 font-medium truncate" title={paymentMode}>

                                    {paymentMode}

                                  </p>

                                  <p className="text-xl font-bold text-gray-900">

                                    {formatCurrency(total)}

                                  </p>

                                  {count > 0 && (

                                    <p className="text-xs text-gray-500 mt-1">

                                      {count} transactions

                                    </p>

                                  )}

                                </div>

                              );

                            }

                          )}

                        </div>

                      ) : (

                        <p className="text-sm text-gray-500 italic">

                          No payment data available for this category

                        </p>

                      )}

                    </div>

                  ))}

                </div>

              </div>

            )}

          </>

        ) : (

          <div className="bg-white rounded-xl shadow-lg p-12 text-center">

            <MdPayment className="mx-auto text-6xl text-gray-300 mb-4" />

            <p className="text-gray-500 text-lg font-medium mb-2">

              No payment data available

            </p>

            <p className="text-gray-400 text-sm mb-4">

              Try adjusting the date range or add some expenses

            </p>

            <Link

              to="/expenses/add"

              className="inline-block px-6 py-3 bg-blue-600 text-white rounded-lg font-medium hover:bg-blue-700 transition-colors"

            >

              Add Expense

            </Link>

          </div>

        )}

      </div>

    </div>

  );

};

export default PaymentAnalysis;

E:\expense\_tracker\frontend\src\pages\analytics\TrendAnalysis.jsx

/\*\*

 \* TrendAnalysis Component

 \* Daily and monthly spending trends with line charts

 \* Backend: GET /api/analytics/daily-trend, /api/analytics/monthly-summary

 \*/

import { useState, useEffect } from 'react';

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

import { toast } from 'react-hot-toast';

import { MdArrowBack, MdRefresh, MdShowChart } from 'react-icons/md';

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

import { formatCurrency } from '../../utils/helpers';

const TrendAnalysis = () => {

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

  const [refreshing, setRefreshing] = useState(false);

  const [dailyTrend, setDailyTrend] = useState([]);

  const [monthlySummary, setMonthlySummary] = useState(null);

  const [days, setDays] = useState(30);

  const fetchTrendData = async (showToast = false) => {

    try {

      if (showToast) {

        setRefreshing(true);

      } else {

        setLoading(true);

      }

      console.log('📊 Fetching trend data for:', days, 'days');

      // ✅ IMPROVED: Use Promise.allSettled for better error handling

      const results = await Promise.allSettled([

        analyticsAPI.getDailyTrend(days),

        analyticsAPI.getMonthlySummary(),

      ]);

      console.log('✅ Trend analysis results:', results);

      // ✅ FIXED: Safe data extraction with multiple fallbacks

      const dailyData = results[0].status === 'fulfilled'

        ? (results[0].value.data?.data || results[0].value.data || [])

        : [];

      const monthlyData = results[1].status === 'fulfilled'

        ? results[1].value.data

        : null;

      console.log('📋 Extracted daily data:', dailyData);

      console.log('📋 Extracted monthly data:', monthlyData);

      // Log which endpoints failed

      if (results[0].status === 'rejected') {

        console.warn('⚠️ Daily trend failed:', results[0].reason?.message);

      }

      if (results[1].status === 'rejected') {

        console.warn('⚠️ Monthly summary failed:', results[1].reason?.message);

      }

      if (Array.isArray(dailyData)) {

        setDailyTrend(dailyData);

      } else {

        console.warn('⚠️ Daily data is not an array:', dailyData);

        setDailyTrend([]);

      }

      // ✅ FIXED: Extract current month data from the new backend structure

      if (monthlyData && monthlyData.data) {

        const currentMonth = new Date().getMonth() + 1;

        const currentMonthData = monthlyData.data.find(

          m => m.month\_number === currentMonth

        );

        setMonthlySummary(currentMonthData || null);

      } else {

        setMonthlySummary(null);

      }

      if (showToast) {

        if (dailyData.length > 0 || monthlyData) {

          toast.success('Data refreshed successfully');

        } else {

          toast.info('No trend data found');

        }

      }

    } catch (error) {

      console.error('❌ Error fetching trend data:', error);

      console.error('Error details:', {

        message: error.message,

        response: error.response?.data,

        status: error.response?.status,

      });

      toast.error(

        error.response?.data?.error ||

        'Failed to load trend analysis'

      );

      setDailyTrend([]);

      setMonthlySummary(null);

    } finally {

      setLoading(false);

      setRefreshing(false);

    }

  };

  useEffect(() => {

    fetchTrendData();

    // eslint-disable-next-line react-hooks/exhaustive-deps

  }, [days]); // Fetch when days changes

  if (loading) {

    return (

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

        <div className="text-center">

          <div className="animate-spin rounded-full h-16 w-16 border-b-2 border-blue-600 mx-auto"></div>

          <p className="mt-4 text-gray-600 font-medium">Loading trend analysis...</p>

        </div>

      </div>

    );

  }

  // ✅ FIXED: Safe calculations with proper number conversion

  const maxTotal = Math.max(...dailyTrend.map((d) => parseFloat(d.total) || 0), 1);

  const totalSpent = dailyTrend.reduce((sum, d) => sum + (parseFloat(d.total) || 0), 0);

  const totalTransactions = dailyTrend.reduce((sum, d) => sum + (parseInt(d.count) || 0), 0);

  const avgDaily = dailyTrend.length > 0 ? totalSpent / dailyTrend.length : 0;

  return (

    <div className="min-h-screen bg-gray-50 p-6">

      <div className="max-w-7xl mx-auto">

        {/\* Header \*/}

        <div className="mb-6">

          <Link

            to="/analytics"

            className="inline-flex items-center gap-2 text-blue-600 hover:text-blue-700 mb-4 font-medium"

          >

            <MdArrowBack className="text-xl" />

            Back to Analytics

          </Link>

          <div className="flex flex-col md:flex-row md:items-center md:justify-between">

            <div>

              <h1 className="text-3xl font-bold text-gray-900">Trend Analysis</h1>

              <p className="mt-1 text-sm text-gray-600">

                Track spending trends over time

              </p>

            </div>

            <button

              onClick={() => fetchTrendData(true)}

              disabled={refreshing}

              className="mt-4 md:mt-0 flex items-center gap-2 px-4 py-2 bg-white rounded-lg shadow-sm text-gray-700 hover:bg-gray-50 transition-colors disabled:opacity-50 disabled:cursor-not-allowed"

            >

              <MdRefresh className={`text-xl ${refreshing ? 'animate-spin' : ''}`} />

              {refreshing ? 'Refreshing...' : 'Refresh'}

            </button>

          </div>

        </div>

        {/\* Period Selector \*/}

        <div className="bg-white rounded-xl shadow-lg p-6 mb-6">

          <h3 className="text-lg font-bold text-gray-900 mb-4">Time Period</h3>

          <div className="flex flex-wrap gap-3">

            <button

              onClick={() => setDays(7)}

              className={`px-6 py-2 rounded-lg font-medium transition-colors ${

                days === 7

                  ? 'bg-blue-600 text-white'

                  : 'bg-gray-100 text-gray-700 hover:bg-gray-200'

              }`}

            >

              Last 7 Days

            </button>

            <button

              onClick={() => setDays(30)}

              className={`px-6 py-2 rounded-lg font-medium transition-colors ${

                days === 30

                  ? 'bg-blue-600 text-white'

                  : 'bg-gray-100 text-gray-700 hover:bg-gray-200'

              }`}

            >

              Last 30 Days

            </button>

            <button

              onClick={() => setDays(90)}

              className={`px-6 py-2 rounded-lg font-medium transition-colors ${

                days === 90

                  ? 'bg-blue-600 text-white'

                  : 'bg-gray-100 text-gray-700 hover:bg-gray-200'

              }`}

            >

              Last 90 Days

            </button>

          </div>

        </div>

        {/\* Summary Cards \*/}

        <div className="grid grid-cols-1 md:grid-cols-4 gap-6 mb-6">

          <div className="bg-white rounded-xl shadow-lg p-6">

            <p className="text-sm text-gray-600 mb-2">Total Spent</p>

            <p className="text-3xl font-bold text-blue-600">

              {formatCurrency(totalSpent)}

            </p>

            <p className="text-xs text-gray-500 mt-2">

              Last {days} days

            </p>

          </div>

          <div className="bg-white rounded-xl shadow-lg p-6">

            <p className="text-sm text-gray-600 mb-2">Daily Average</p>

            <p className="text-3xl font-bold text-purple-600">

              {formatCurrency(avgDaily)}

            </p>

            <p className="text-xs text-gray-500 mt-2">

              Per day

            </p>

          </div>

          <div className="bg-white rounded-xl shadow-lg p-6">

            <p className="text-sm text-gray-600 mb-2">Transactions</p>

            <p className="text-3xl font-bold text-green-600">

              {totalTransactions}

            </p>

            <p className="text-xs text-gray-500 mt-2">

              Total count

            </p>

          </div>

          <div className="bg-white rounded-xl shadow-lg p-6">

            <p className="text-sm text-gray-600 mb-2">Days with Data</p>

            <p className="text-3xl font-bold text-orange-600">

              {dailyTrend.filter(d => (d.total || 0) > 0).length}

            </p>

            <p className="text-xs text-gray-500 mt-2">

              Out of {dailyTrend.length}

            </p>

          </div>

        </div>

        {/\* Daily Trend Chart \*/}

        {dailyTrend.length > 0 ? (

          <div className="bg-white rounded-xl shadow-lg p-6 mb-6">

            <h3 className="text-xl font-bold text-gray-900 mb-6">

              Daily Spending Trend

            </h3>

            {/\* Bar Chart \*/}

            <div className="h-80 mb-4">

              <div className="h-full flex items-end justify-between gap-1">

                {dailyTrend.map((day, index) => {

                  const total = parseFloat(day.total) || 0;

                  const count = parseInt(day.count) || 0;

                  const height = maxTotal > 0 ? (total / maxTotal) \* 100 : 0;

                  return (

                    <div key={index} className="flex-1 flex flex-col items-center group relative min-w-0">

                      {/\* Tooltip \*/}

                      <div className="absolute bottom-full mb-2 opacity-0 group-hover:opacity-100 transition-opacity bg-gray-900 text-white text-xs rounded py-2 px-3 whitespace-nowrap pointer-events-none z-10">

                        <div className="font-semibold">{day.dayname || day.date}</div>

                        <div>{formatCurrency(total)}</div>

                        <div>{count} transactions</div>

                        {count > 0 && (

                          <div className="text-gray-300">

                            Avg: {formatCurrency(total / count)}

                          </div>

                        )}

                      </div>

                      {/\* Bar Container \*/}

                      <div className="w-full h-full flex items-end">

                        <div

                          className="w-full bg-gradient-to-t from-blue-500 to-blue-400 rounded-t-lg transition-all duration-300 hover:from-blue-600 hover:to-blue-500 cursor-pointer"

                          style={{

                            height: `${height}%`,

                            minHeight: height > 0 ? '4px' : '0',

                          }}

                        ></div>

                      </div>

                      {/\* Day Label \*/}

                      <p className="text-xs text-gray-600 mt-2 text-center truncate w-full">

                        {day.dayname ? day.dayname.slice(0, 3) : (day.date?.slice(-5) || '')}

                      </p>

                    </div>

                  );

                })}

              </div>

            </div>

            {/\* Y-axis labels \*/}

            <div className="flex justify-between text-xs text-gray-500 border-t pt-2">

              <span>{formatCurrency(0)}</span>

              <span>{formatCurrency(maxTotal / 2)}</span>

              <span>{formatCurrency(maxTotal)}</span>

            </div>

            {/\* Legend \*/}

            <div className="mt-4 flex items-center justify-center gap-6 text-sm text-gray-600">

              <div className="flex items-center gap-2">

                <div className="w-4 h-4 bg-gradient-to-t from-blue-500 to-blue-400 rounded"></div>

                <span>Daily Spending</span>

              </div>

              <div className="flex items-center gap-2">

                <div className="w-4 h-4 border-2 border-blue-400 rounded"></div>

                <span>Hover for details</span>

              </div>

            </div>

          </div>

        ) : (

          <div className="bg-white rounded-xl shadow-lg p-12 text-center mb-6">

            <MdShowChart className="mx-auto text-6xl text-gray-300 mb-4" />

            <p className="text-gray-500 text-lg font-medium mb-2">

              No trend data available

            </p>

            <p className="text-gray-400 text-sm mb-4">

              Add some expenses to see spending trends

            </p>

            <Link

              to="/expenses/add"

              className="inline-block px-6 py-3 bg-blue-600 text-white rounded-lg font-medium hover:bg-blue-700 transition-colors"

            >

              Add Expense

            </Link>

          </div>

        )}

        {/\* Monthly Summary \*/}

        {monthlySummary && (

          <div className="bg-white rounded-xl shadow-lg p-6">

            <h3 className="text-xl font-bold text-gray-900 mb-6">

              Current Month Summary

            </h3>

            <div className="grid grid-cols-1 md:grid-cols-4 gap-6">

              <div className="text-center p-4 bg-blue-50 rounded-lg border border-blue-200">

                <p className="text-sm text-blue-600 font-medium mb-2">Month</p>

                <p className="text-2xl font-bold text-gray-900">

                  {monthlySummary.month || 'N/A'}

                </p>

              </div>

              <div className="text-center p-4 bg-purple-50 rounded-lg border border-purple-200">

                <p className="text-sm text-purple-600 font-medium mb-2">

                  Total Spent

                </p>

                <p className="text-2xl font-bold text-gray-900">

                  {formatCurrency(parseFloat(monthlySummary.total) || 0)}

                </p>

              </div>

              <div className="text-center p-4 bg-green-50 rounded-lg border border-green-200">

                <p className="text-sm text-green-600 font-medium mb-2">

                  Transactions

                </p>

                <p className="text-2xl font-bold text-gray-900">

                  {monthlySummary.count || 0}

                </p>

              </div>

              <div className="text-center p-4 bg-orange-50 rounded-lg border border-orange-200">

                <p className="text-sm text-orange-600 font-medium mb-2">

                  Average

                </p>

                <p className="text-2xl font-bold text-gray-900">

                  {formatCurrency(parseFloat(monthlySummary.average) || 0)}

                </p>

              </div>

            </div>

          </div>

        )}

      </div>

    </div>

  );

};

export default TrendAnalysis;

E:\expense\_tracker\frontend\src\pages\auth

E:\expense\_tracker\frontend\src\pages\auth\ForgotPassword.jsx

/\*\*

 \* Forgot Password Page Component

 \* Note: Backend doesn't have password reset endpoint yet

 \* This is a placeholder UI that can be connected when backend implements it

 \*/

import { useState } from 'react';

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

import { toast } from 'react-hot-toast';

const ForgotPassword = () => {

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

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

  const [emailSent, setEmailSent] = useState(false);

  const handleSubmit = async (e) => {

    e.preventDefault();

    // Email validation

    const emailRegex = /^[a-zA-Z0-9.\_-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;

    if (!emailRegex.test(email)) {

      toast.error('Please enter a valid email address');

      return;

    }

    setLoading(true);

    try {

      // TODO: Implement backend password reset endpoint

      // await authAPI.forgotPassword({ email });

      // Simulated delay

      await new Promise((resolve) => setTimeout(resolve, 1500));

      setEmailSent(true);

      toast.success('Password reset instructions sent to your email');

    } catch (error) {

      console.error('Forgot password error:', error);

      toast.error('Failed to send reset email. Please try again.');

    } finally {

      setLoading(false);

    }

  };

  if (emailSent) {

    return (

      <div className="min-h-screen flex items-center justify-center bg-gradient-to-br from-purple-500 via-pink-500 to-red-500 py-12 px-4 sm:px-6 lg:px-8">

        <div className="max-w-md w-full space-y-8 bg-white rounded-2xl shadow-2xl p-8 text-center">

          <div className="mx-auto flex items-center justify-center h-16 w-16 rounded-full bg-green-100">

            <svg className="h-8 w-8 text-green-600" fill="none" viewBox="0 0 24 24" stroke="currentColor">

              <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M5 13l4 4L19 7" />

            </svg>

          </div>

          <div>

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

              Check Your Email

            </h2>

            <p className="mt-4 text-gray-600">

              We've sent password reset instructions to <strong>{email}</strong>

            </p>

            <p className="mt-2 text-sm text-gray-500">

              Please check your inbox and follow the link to reset your password.

            </p>

          </div>

          <div className="space-y-3">

            <Link

              to="/login"

              className="w-full inline-flex justify-center py-3 px-4 border border-transparent rounded-lg shadow-sm text-sm font-medium text-white bg-gradient-to-r from-blue-600 to-purple-600 hover:from-blue-700 hover:to-purple-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-blue-500 transition-all"

            >

              Back to Login

            </Link>

            <button

              onClick={() => setEmailSent(false)}

              className="w-full inline-flex justify-center py-3 px-4 border border-gray-300 rounded-lg shadow-sm text-sm font-medium text-gray-700 bg-white hover:bg-gray-50 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-blue-500 transition-all"

            >

              Try Another Email

            </button>

          </div>

        </div>

      </div>

    );

  }

  return (

    <div className="min-h-screen flex items-center justify-center bg-gradient-to-br from-purple-500 via-pink-500 to-red-500 py-12 px-4 sm:px-6 lg:px-8">

      <div className="max-w-md w-full space-y-8 bg-white rounded-2xl shadow-2xl p-8">

        {/\* Header \*/}

        <div>

          <h2 className="text-center text-4xl font-extrabold text-gray-900">

            Forgot Password?

          </h2>

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

            No worries! Enter your email and we'll send you reset instructions.

          </p>

        </div>

        {/\* Form \*/}

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

          <div>

            <label htmlFor="email" className="block text-sm font-medium text-gray-700 mb-1">

              Email Address

            </label>

            <input

              id="email"

              name="email"

              type="email"

              autoComplete="email"

              required

              value={email}

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

              className="appearance-none relative block w-full px-4 py-3 border border-gray-300 placeholder-gray-500 text-gray-900 rounded-lg focus:outline-none focus:ring-2 focus:ring-purple-500 focus:border-purple-500 sm:text-sm transition-all"

              placeholder="you@example.com"

            />

          </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-lg text-white bg-gradient-to-r from-purple-600 to-pink-600 hover:from-purple-700 hover:to-pink-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-purple-500 disabled:opacity-50 disabled:cursor-not-allowed transition-all transform hover:scale-[1.02]"

            >

              {loading ? (

                <span className="flex items-center">

                  <svg className="animate-spin -ml-1 mr-3 h-5 w-5 text-white" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24">

                    <circle className="opacity-25" cx="12" cy="12" r="10" stroke="currentColor" strokeWidth="4"></circle>

                    <path className="opacity-75" fill="currentColor" d="M4 12a8 8 0 018-8V0C5.373 0 0 5.373 0 12h4zm2 5.291A7.962 7.962 0 014 12H0c0 3.042 1.135 5.824 3 7.938l3-2.647z"></path>

                  </svg>

                  Sending...

                </span>

              ) : (

                'Send Reset Link'

              )}

            </button>

          </div>

          <div className="text-center">

            <Link

              to="/login"

              className="text-sm font-medium text-purple-600 hover:text-purple-500 transition-colors inline-flex items-center"

            >

              <svg className="w-4 h-4 mr-1" fill="none" viewBox="0 0 24 24" stroke="currentColor">

                <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M10 19l-7-7m0 0l7-7m-7 7h18" />

              </svg>

              Back to Login

            </Link>

          </div>

        </form>

      </div>

    </div>

  );

};

export default ForgotPassword;

E:\expense\_tracker\frontend\src\pages\auth\Login.jsx

/\*\*

 \* Login Component

 \* User login page with email and password authentication

 \*/

import { useState } from 'react';

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

import { toast } from 'react-hot-toast';

import { MdEmail, MdLock, MdVisibility, MdVisibilityOff } from 'react-icons/md';

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

import { setAuthToken, setUserData } from '../../utils/auth';

const Login = () => {

  const location = useLocation();

  const navigate = useNavigate();

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

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

  const [formData, setFormData] = useState({

    email: '',

    password: '',

  });

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

  const handleChange = (e) => {

    const { name, value } = e.target;

    setFormData((prev) => ({

      ...prev,

      [name]: value,

    }));

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    console.log('🔐 === LOGIN PROCESS START ===');

    console.log('1. Form submitted');

    console.log('   Email:', formData.email);

    // Validation

    if (!formData.email || !formData.password) {

      toast.error('Please fill in all fields');

      return;

    }

    if (!/\S+@\S+\.\S+/.test(formData.email)) {

      toast.error('Please enter a valid email');

      return;

    }

    setLoading(true);

    try {

      console.log('2. Calling login API...');

      const response = await authAPI.login({

        email: formData.email,

        password: formData.password,

      });

      console.log('3. ✅ API Response received');

      console.log('   Full response:', response.data);

      const { access\_token, user } = response.data;

      if (!access\_token) {

        throw new Error('No access token in response');

      }

      console.log('4. 💾 Saving credentials...');

      console.log('   Token length:', access\_token.length);

      console.log('   User:', user.email);

      setAuthToken(access\_token);

      setUserData(user);

      const verifyToken = localStorage.getItem('expense\_tracker\_token');

      const verifyUser = localStorage.getItem('expense\_tracker\_user');

      console.log('5. ✅ Verification:');

      console.log('   Token saved:', !!verifyToken);

      console.log('   User saved:', !!verifyUser);

      if (!verifyToken || !verifyUser) {

        console.error('❌ CRITICAL: localStorage save failed!');

        throw new Error('Failed to save login data');

      }

      toast.success('Login successful!');

      console.log('6. 🔄 Redirecting to:', from);

      console.log('   Current URL:', window.location.href);

      // Immediate redirect using navigate

      navigate(from);

    } catch (error) {

      console.error('❌ === LOGIN ERROR ===');

      console.error('Error type:', error.constructor.name);

      console.error('Error message:', error.message);

      console.error('Error response:', error.response?.data);

      setLoading(false);

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

        toast.error('Invalid email or password');

      } else if (error.response?.status === 400) {

        toast.error(error.response?.data?.message || 'Invalid credentials');

      } else if (error.response?.status === 500) {

        toast.error('Server error. Please try again.');

      } else if (error.message.includes('Network Error')) {

        toast.error('Cannot connect to server');

      } else {

        toast.error(error.message || 'Login failed');

      }

    }

  };

  return (

    <div className="w-full max-w-md px-6">

      <div className="bg-white rounded-2xl shadow-2xl p-8">

        <div className="text-center mb-8">

          <h2 className="text-3xl font-bold text-gray-900 mb-2">Welcome Back</h2>

          <p className="text-gray-600">Sign in to continue to Expense Tracker</p>

        </div>

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

          <div>

            <label

              htmlFor="email"

              className="block text-sm font-medium text-gray-700 mb-2"

            >

              Email Address

            </label>

            <div className="relative">

              <div className="absolute inset-y-0 left-0 pl-3 flex items-center pointer-events-none">

                <MdEmail className="text-gray-400 text-xl" />

              </div>

              <input

                type="email"

                id="email"

                name="email"

                value={formData.email}

                onChange={handleChange}

                placeholder="Enter your email"

                disabled={loading}

                className="w-full pl-10 pr-4 py-3 border border-gray-300 rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500 disabled:bg-gray-100 disabled:cursor-not-allowed transition-colors"

                autoComplete="email"

                required

              />

            </div>

          </div>

          <div>

            <label

              htmlFor="password"

              className="block text-sm font-medium text-gray-700 mb-2"

            >

              Password

            </label>

            <div className="relative">

              <div className="absolute inset-y-0 left-0 pl-3 flex items-center pointer-events-none">

                <MdLock className="text-gray-400 text-xl" />

              </div>

              <input

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

                id="password"

                name="password"

                value={formData.password}

                onChange={handleChange}

                placeholder="Enter your password"

                disabled={loading}

                className="w-full pl-10 pr-12 py-3 border border-gray-300 rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500 disabled:bg-gray-100 disabled:cursor-not-allowed transition-colors"

                autoComplete="current-password"

                required

              />

              <button

                type="button"

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

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

                tabIndex="-1"

              >

                {showPassword ? (

                  <MdVisibilityOff className="text-gray-400 text-xl hover:text-gray-600 transition-colors" />

                ) : (

                  <MdVisibility className="text-gray-400 text-xl hover:text-gray-600 transition-colors" />

                )}

              </button>

            </div>

          </div>

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

            <div className="flex items-center">

              <input

                id="remember-me"

                name="remember-me"

                type="checkbox"

                className="h-4 w-4 text-blue-600 focus:ring-blue-500 border-gray-300 rounded"

              />

              <label

                htmlFor="remember-me"

                className="ml-2 block text-sm text-gray-700"

              >

                Remember me

              </label>

            </div>

            <Link

              to="/forgot-password"

              className="text-sm font-medium text-blue-600 hover:text-blue-500 transition-colors"

            >

              Forgot password?

            </Link>

          </div>

          <button

            type="submit"

            disabled={loading}

            className="w-full flex items-center justify-center px-6 py-3 bg-gradient-to-r from-blue-600 to-purple-600 text-white rounded-lg font-semibold hover:from-blue-700 hover:to-purple-700 focus:outline-none focus:ring-2 focus:ring-blue-500 focus:ring-offset-2 disabled:opacity-50 disabled:cursor-not-allowed transition-all shadow-lg hover:shadow-xl"

          >

            {loading ? (

              <>

                <svg

                  className="animate-spin -ml-1 mr-3 h-5 w-5 text-white"

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

                  fill="none"

                  viewBox="0 0 24 24"

                >

                  <circle

                    className="opacity-25"

                    cx="12"

                    cy="12"

                    r="10"

                    stroke="currentColor"

                    strokeWidth="4"

                  ></circle>

                  <path

                    className="opacity-75"

                    fill="currentColor"

                    d="M4 12a8 8 0 018-8V0C5.373 0 0 5.373 0 12h4zm2 5.291A7.962 7.962 0 014 12H0c0 3.042 1.135 5.824 3 7.938l3-2.647z"

                  ></path>

                </svg>

                Signing in...

              </>

            ) : (

              'Sign In'

            )}

          </button>

        </form>

        <div className="mt-6">

          <div className="relative">

            <div className="absolute inset-0 flex items-center">

              <div className="w-full border-t border-gray-300"></div>

            </div>

            <div className="relative flex justify-center text-sm">

              <span className="px-2 bg-white text-gray-500">

                Don't have an account?

              </span>

            </div>

          </div>

        </div>

        <div className="mt-6 text-center">

          <Link

            to="/register"

            className="text-blue-600 hover:text-blue-500 font-medium transition-colors"

          >

            Create a new account

          </Link>

        </div>

      </div>

      <div className="mt-8 text-center">

        <p className="text-sm text-gray-600">

          By signing in, you agree to our{' '}

          <a href="#" className="text-blue-600 hover:underline">

            Terms of Service

          </a>{' '}

          and{' '}

          <a href="#" className="text-blue-600 hover:underline">

            Privacy Policy

          </a>

        </p>

      </div>

    </div>

  );

};

export default Login;

E:\expense\_tracker\frontend\src\pages\auth\Register.jsx

/\*\*

 \* Register Page Component

 \* Handles new user registration

 \* Backend: POST /api/auth/register

 \*/

import { useState } from 'react';

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

import { toast } from 'react-hot-toast';

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

const Register = () => {

  const navigate = useNavigate();

  const [formData, setFormData] = useState({

    email: '',

    password: '',

    confirmPassword: '',

    fullname: '',

  });

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

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

  const handleChange = (e) => {

    const { name, value } = e.target;

    setFormData((prev) => ({

      ...prev,

      [name]: value,

    }));

  };

  const validateForm = () => {

    // Email validation

    const emailRegex = /^[a-zA-Z0-9.\_-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;

    if (!emailRegex.test(formData.email)) {

      toast.error('Invalid email format');

      return false;

    }

    // Password validation - Backend requires minimum 6 characters

    if (formData.password.length < 6) {

      toast.error('Password must be at least 6 characters long');

      return false;

    }

    // Confirm password match

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

      toast.error('Passwords do not match');

      return false;

    }

    return true;

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    if (!validateForm()) {

      return;

    }

    setLoading(true);

    try {

      // Backend expects: { email, password, full\_name (optional) }

      const response = await authAPI.register({

        email: formData.email.toLowerCase().trim(),

        password: formData.password,

        full\_name: formData.fullname.trim() || undefined,

      });

      // Backend returns: { message, user }

      toast.success('Registration successful! Please log in.');

      // Redirect to login page

      navigate('/login');

    } catch (error) {

      console.error('Registration error:', error);

      // Backend returns 409 if user already exists

      if (error.response?.status === 409) {

        toast.error('User with this email already exists');

      } else if (error.response?.status === 400) {

        toast.error(error.response?.data?.error || 'Invalid registration data');

      } else {

        toast.error(error.response?.data?.error || 'Registration failed. Please try again.');

      }

    } finally {

      setLoading(false);

    }

  };

  return (

    <div className="min-h-screen flex items-center justify-center bg-gradient-to-br from-green-500 via-teal-500 to-blue-500 py-12 px-4 sm:px-6 lg:px-8">

      <div className="max-w-md w-full space-y-8 bg-white rounded-2xl shadow-2xl p-8">

        {/\* Header \*/}

        <div>

          <h2 className="text-center text-4xl font-extrabold text-gray-900">

            Create Account

          </h2>

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

            Join Expense Tracker and manage your finances

          </p>

        </div>

        {/\* Registration Form \*/}

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

          <div className="space-y-4">

            {/\* Full Name Input (Optional) \*/}

            <div>

              <label htmlFor="fullname" className="block text-sm font-medium text-gray-700 mb-1">

                Full Name <span className="text-gray-400">(Optional)</span>

              </label>

              <input

                id="fullname"

                name="fullname"

                type="text"

                autoComplete="name"

                value={formData.fullname}

                onChange={handleChange}

                maxLength={100}

                className="appearance-none relative block w-full px-4 py-3 border border-gray-300 placeholder-gray-500 text-gray-900 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500 focus:border-green-500 sm:text-sm transition-all"

                placeholder="John Doe"

              />

            </div>

            {/\* Email Input \*/}

            <div>

              <label htmlFor="email" className="block text-sm font-medium text-gray-700 mb-1">

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

              </label>

              <input

                id="email"

                name="email"

                type="email"

                autoComplete="email"

                required

                value={formData.email}

                onChange={handleChange}

                maxLength={120}

                className="appearance-none relative block w-full px-4 py-3 border border-gray-300 placeholder-gray-500 text-gray-900 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500 focus:border-green-500 sm:text-sm transition-all"

                placeholder="you@example.com"

              />

            </div>

            {/\* Password Input \*/}

            <div>

              <label htmlFor="password" className="block text-sm font-medium text-gray-700 mb-1">

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

              </label>

              <div className="relative">

                <input

                  id="password"

                  name="password"

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

                  autoComplete="new-password"

                  required

                  value={formData.password}

                  onChange={handleChange}

                  className="appearance-none relative block w-full px-4 py-3 border border-gray-300 placeholder-gray-500 text-gray-900 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500 focus:border-green-500 sm:text-sm transition-all"

                  placeholder="Minimum 6 characters"

                />

                <button

                  type="button"

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

                  className="absolute inset-y-0 right-0 pr-3 flex items-center text-sm text-gray-600 hover:text-gray-800"

                >

                  {showPassword ? '🙈' : '👁️'}

                </button>

              </div>

              <p className="mt-1 text-xs text-gray-500">

                Must be at least 6 characters long

              </p>

            </div>

            {/\* Confirm Password Input \*/}

            <div>

              <label htmlFor="confirmPassword" className="block text-sm font-medium text-gray-700 mb-1">

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

              </label>

              <input

                id="confirmPassword"

                name="confirmPassword"

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

                autoComplete="new-password"

                required

                value={formData.confirmPassword}

                onChange={handleChange}

                className="appearance-none relative block w-full px-4 py-3 border border-gray-300 placeholder-gray-500 text-gray-900 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500 focus:border-green-500 sm:text-sm transition-all"

                placeholder="Re-enter your password"

              />

            </div>

          </div>

          {/\* Submit Button \*/}

          <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-lg text-white bg-gradient-to-r from-green-600 to-teal-600 hover:from-green-700 hover:to-teal-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-green-500 disabled:opacity-50 disabled:cursor-not-allowed transition-all transform hover:scale-[1.02]"

            >

              {loading ? (

                <span className="flex items-center">

                  <svg className="animate-spin -ml-1 mr-3 h-5 w-5 text-white" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24">

                    <circle className="opacity-25" cx="12" cy="12" r="10" stroke="currentColor" strokeWidth="4"></circle>

                    <path className="opacity-75" fill="currentColor" d="M4 12a8 8 0 018-8V0C5.373 0 0 5.373 0 12h4zm2 5.291A7.962 7.962 0 014 12H0c0 3.042 1.135 5.824 3 7.938l3-2.647z"></path>

                  </svg>

                  Creating account...

                </span>

              ) : (

                'Create Account'

              )}

            </button>

          </div>

          {/\* Login Link \*/}

          <div className="text-center">

            <p className="text-sm text-gray-600">

              Already have an account?{' '}

              <Link

                to="/login"

                className="font-medium text-green-600 hover:text-green-500 transition-colors"

              >

                Sign in here

              </Link>

            </p>

          </div>

        </form>

      </div>

    </div>

  );

};

export default Register;

E:\expense\_tracker\frontend\src\pages\dashboard

/\*\*

 \* Dashboard Component

 \* Main dashboard displaying expense overview, analytics, and recent activities

 \*/

import { useState, useEffect } from 'react';

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

import { toast } from 'react-hot-toast';

import {

  MdTrendingUp,

  MdAccountBalanceWallet,

  MdBarChart,

  MdAttachMoney,

  MdCalendarToday,

  MdArrowForward,

  MdRefresh

} from 'react-icons/md';

import { analyticsAPI, expenseAPI } from '../../services/api';

import { formatCurrency, formatDate } from '../../utils/helpers';

const Dashboard = () => {

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

  const [refreshing, setRefreshing] = useState(false);

  const [dashboardData, setDashboardData] = useState({

    summary: {

      totalExpenses: 0,

      expenseCount: 0,

      dailyAverage: 0,

      period: 7,

    },

    categoryBreakdown: [],

    paymentBreakdown: [],

    dailyTrend: [],

    recentExpenses: [],

    topCategory: null,

    mostUsedPayment: null,

  });

  const [dateFilter, setDateFilter] = useState({

    days: 7,

    label: 'Last 7 Days'

  });

  // ✅ FIXED: Fetch with graceful error handling

  const fetchDashboardData = async (showToast = false) => {

    try {

      if (showToast) {

        setRefreshing(true);

      } else {

        setLoading(true);

      }

      console.log('📊 Fetching dashboard data for:', dateFilter.label);

      const today = new Date();

      const startDate = new Date(today);

      startDate.setDate(today.getDate() - dateFilter.days);

      const startDateStr = startDate.toISOString().split('T')[0];

      const endDateStr = today.toISOString().split('T')[0];

      console.log('📅 Date range:', startDateStr, 'to', endDateStr);

      // ✅ FIXED: Use Promise.allSettled instead of Promise.all

      // This prevents one failing endpoint from crashing everything

      const results = await Promise.allSettled([

        analyticsAPI.getCategoryBreakdown(startDateStr, endDateStr),

        analyticsAPI.getPaymentBreakdown(startDateStr, endDateStr),

        analyticsAPI.getDailyTrend(dateFilter.days),

        expenseAPI.getExpenses({

          start\_date: startDateStr,

          end\_date: endDateStr,

          per\_page: 5

        }),

      ]);

      console.log('✅ Dashboard API calls completed');

      // ✅ FIXED: Extract data with safe fallbacks

      const categories = results[0].status === 'fulfilled'

        ? (results[0].value?.data?.data || [])

        : [];

      const payments = results[1].status === 'fulfilled'

        ? (results[1].value?.data?.data || [])

        : [];

      const dailyData = results[2].status === 'fulfilled'

        ? (results[2].value?.data?.data || [])

        : [];

      const expenses = results[3].status === 'fulfilled'

        ? (results[3].value?.data?.expenses || results[3].value?.data?.data || [])

        : [];

      // ✅ Log which endpoints failed

      const endpointNames = ['Categories', 'Payments', 'Daily Trend', 'Expenses'];

      results.forEach((result, index) => {

        if (result.status === 'rejected') {

          console.warn(`⚠️ ${endpointNames[index]} endpoint failed:`, result.reason?.message);

        } else {

          console.log(`✅ ${endpointNames[index]} loaded successfully`);

        }

      });

      // Process data

      const totalExpenses = categories.reduce((sum, cat) => sum + (cat.total || 0), 0);

      const expenseCount = categories.reduce((sum, cat) => sum + (cat.count || 0), 0);

      const topCategory = categories.length > 0 ? categories[0] : null;

      const mostUsedPayment = payments.length > 0

        ? payments.reduce((prev, current) =>

            ((prev.count || 0) > (current.count || 0)) ? prev : current

          )

        : null;

      const dailyAverage = dateFilter.days > 0

        ? totalExpenses / dateFilter.days

        : 0;

      setDashboardData({

        summary: {

          totalExpenses,

          expenseCount,

          dailyAverage,

          period: dateFilter.days,

        },

        categoryBreakdown: categories.slice(0, 5),

        paymentBreakdown: payments.slice(0, 5),

        dailyTrend: dailyData,

        recentExpenses: expenses,

        topCategory,

        mostUsedPayment,

      });

      if (showToast) {

        toast.success('Dashboard refreshed');

      }

    } catch (error) {

      console.error('❌ Critical error fetching dashboard data:', error);

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

        toast.error('Session expired. Please login again.');

        window.location.href = '/login';

        return;

      }

      // ✅ Set empty data so dashboard still renders

      setDashboardData({

        summary: {

          totalExpenses: 0,

          expenseCount: 0,

          dailyAverage: 0,

          period: dateFilter.days,

        },

        categoryBreakdown: [],

        paymentBreakdown: [],

        dailyTrend: [],

        recentExpenses: [],

        topCategory: null,

        mostUsedPayment: null,

      });

      toast.error('Unable to load some dashboard data', {

        duration: 3000,

      });

    } finally {

      setLoading(false);

      setRefreshing(false);

    }

  };

  useEffect(() => {

    fetchDashboardData();

    // eslint-disable-next-line react-hooks/exhaustive-deps

  }, [dateFilter.days]);

  const handleRefresh = () => {

    fetchDashboardData(true);

  };

  const handleDateFilterChange = (days, label) => {

    setDateFilter({ days, label });

  };

  if (loading) {

    return (

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

        <div className="text-center">

          <div className="animate-spin rounded-full h-16 w-16 border-b-4 border-blue-600 mx-auto"></div>

          <p className="mt-4 text-gray-600 font-medium">Loading dashboard...</p>

        </div>

      </div>

    );

  }

  return (

    <div className="min-h-screen bg-gray-50 p-6">

      <div className="max-w-7xl mx-auto">

        {/\* Header \*/}

        <div className="flex flex-col md:flex-row md:items-center md:justify-between mb-8">

          <div>

            <h1 className="text-3xl font-bold text-gray-900">Dashboard</h1>

            <p className="mt-1 text-sm text-gray-600">

              Overview of your expenses - {dateFilter.label}

            </p>

          </div>

          <div className="mt-4 md:mt-0 flex flex-wrap gap-3">

            {/\* Date Filter Buttons \*/}

            <div className="flex gap-2 bg-white rounded-lg p-1 shadow-sm">

              <button

                onClick={() => handleDateFilterChange(7, 'Last 7 Days')}

                className={`px-4 py-2 rounded-md text-sm font-medium transition-colors ${

                  dateFilter.days === 7

                    ? 'bg-blue-600 text-white'

                    : 'text-gray-600 hover:bg-gray-100'

                }`}

              >

                7D

              </button>

              <button

                onClick={() => handleDateFilterChange(30, 'Last 30 Days')}

                className={`px-4 py-2 rounded-md text-sm font-medium transition-colors ${

                  dateFilter.days === 30

                    ? 'bg-blue-600 text-white'

                    : 'text-gray-600 hover:bg-gray-100'

                }`}

              >

                30D

              </button>

              <button

                onClick={() => handleDateFilterChange(90, 'Last 90 Days')}

                className={`px-4 py-2 rounded-md text-sm font-medium transition-colors ${

                  dateFilter.days === 90

                    ? 'bg-blue-600 text-white'

                    : 'text-gray-600 hover:bg-gray-100'

                }`}

              >

                90D

              </button>

            </div>

            {/\* Refresh Button \*/}

            <button

              onClick={handleRefresh}

              disabled={refreshing}

              className="flex items-center gap-2 px-4 py-2 bg-white rounded-lg shadow-sm text-gray-700 hover:bg-gray-50 transition-colors disabled:opacity-50 disabled:cursor-not-allowed"

            >

              <MdRefresh className={`text-xl ${refreshing ? 'animate-spin' : ''}`} />

              Refresh

            </button>

          </div>

        </div>

        {/\* Summary Cards \*/}

        <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6 mb-8">

          {/\* Total Expenses Card \*/}

          <div className="bg-gradient-to-br from-blue-500 to-blue-600 rounded-xl shadow-lg p-6 text-white">

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

              <div>

                <p className="text-blue-100 text-sm font-medium">Total Expenses</p>

                <h3 className="text-3xl font-bold mt-2">

                  {formatCurrency(dashboardData.summary.totalExpenses)}

                </h3>

              </div>

              <div className="bg-blue-400 bg-opacity-50 rounded-full p-3">

                <MdAccountBalanceWallet className="text-3xl" />

              </div>

            </div>

            <p className="text-blue-100 text-xs mt-3">

              {dateFilter.label}

            </p>

          </div>

          {/\* Transaction Count Card \*/}

          <div className="bg-gradient-to-br from-purple-500 to-purple-600 rounded-xl shadow-lg p-6 text-white">

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

              <div>

                <p className="text-purple-100 text-sm font-medium">Transactions</p>

                <h3 className="text-3xl font-bold mt-2">

                  {dashboardData.summary.expenseCount}

                </h3>

              </div>

              <div className="bg-purple-400 bg-opacity-50 rounded-full p-3">

                <MdBarChart className="text-3xl" />

              </div>

            </div>

            <p className="text-purple-100 text-xs mt-3">

              Total count

            </p>

          </div>

          {/\* Daily Average Card \*/}

          <div className="bg-gradient-to-br from-green-500 to-green-600 rounded-xl shadow-lg p-6 text-white">

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

              <div>

                <p className="text-green-100 text-sm font-medium">Daily Average</p>

                <h3 className="text-3xl font-bold mt-2">

                  {formatCurrency(dashboardData.summary.dailyAverage)}

                </h3>

              </div>

              <div className="bg-green-400 bg-opacity-50 rounded-full p-3">

                <MdTrendingUp className="text-3xl" />

              </div>

            </div>

            <p className="text-green-100 text-xs mt-3">

              Per day

            </p>

          </div>

          {/\* Top Category Card \*/}

          <div className="bg-gradient-to-br from-orange-500 to-orange-600 rounded-xl shadow-lg p-6 text-white">

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

              <div>

                <p className="text-orange-100 text-sm font-medium">Top Category</p>

                <h3 className="text-xl font-bold mt-2 truncate">

                  {dashboardData.topCategory?.category || 'N/A'}

                </h3>

              </div>

              <div className="bg-orange-400 bg-opacity-50 rounded-full p-3">

                <MdAttachMoney className="text-3xl" />

              </div>

            </div>

            <p className="text-orange-100 text-xs mt-3">

              {dashboardData.topCategory

                ? formatCurrency(dashboardData.topCategory.total)

                : 'No data'}

            </p>

          </div>

        </div>

        {/\* Charts and Lists Section \*/}

        <div className="grid grid-cols-1 lg:grid-cols-3 gap-6">

          {/\* Daily Trend Chart \*/}

          <div className="lg:col-span-2 bg-white rounded-xl shadow-lg p-6">

            <div className="flex items-center justify-between mb-6">

              <h2 className="text-xl font-bold text-gray-900">Daily Spending Trend</h2>

              <Link

                to="/analytics"

                className="text-blue-600 hover:text-blue-700 text-sm font-medium flex items-center gap-1"

              >

                View Details <MdArrowForward />

              </Link>

            </div>

            {dashboardData.dailyTrend.length > 0 ? (

              <div className="h-64">

                <div className="flex items-end justify-between h-full gap-2">

                  {dashboardData.dailyTrend.map((item, index) => {

                    const maxTotal = Math.max(...dashboardData.dailyTrend.map(d => d.total || 0));

                    const height = maxTotal > 0 ? ((item.total || 0) / maxTotal) \* 100 : 0;

                    return (

                      <div key={index} className="flex-1 flex flex-col items-center min-w-0">

                        <div className="w-full bg-gray-100 rounded-t-lg overflow-hidden relative group" style={{ height: '200px' }}>

                          <div

                            className="bg-gradient-to-t from-blue-500 to-blue-400 transition-all duration-300 hover:from-blue-600 hover:to-blue-500 absolute bottom-0 w-full"

                            style={{ height: `${height}%`, minHeight: height > 0 ? '4px' : '0' }}

                          >

                            <div className="absolute -top-12 left-1/2 transform -translate-x-1/2 bg-gray-900 text-white text-xs rounded py-1 px-2 opacity-0 group-hover:opacity-100 transition-opacity whitespace-nowrap z-10 pointer-events-none">

                              {formatCurrency(item.total || 0)}

                            </div>

                          </div>

                        </div>

                        <p className="text-xs text-gray-600 mt-2 text-center truncate w-full">

                          {item.dayname?.slice(0, 3) || item.date}

                        </p>

                      </div>

                    );

                  })}

                </div>

              </div>

            ) : (

              <div className="h-64 flex items-center justify-center text-gray-400">

                <div className="text-center">

                  <MdBarChart className="text-5xl mx-auto mb-2 opacity-50" />

                  <p>No data available for the selected period</p>

                </div>

              </div>

            )}

          </div>

          {/\* Recent Expenses \*/}

          <div className="bg-white rounded-xl shadow-lg p-6">

            <div className="flex items-center justify-between mb-6">

              <h2 className="text-xl font-bold text-gray-900">Recent Expenses</h2>

              <Link

                to="/expenses"

                className="text-blue-600 hover:text-blue-700 text-sm font-medium flex items-center gap-1"

              >

                View All <MdArrowForward />

              </Link>

            </div>

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

              {dashboardData.recentExpenses.length > 0 ? (

                dashboardData.recentExpenses.map((expense) => (

                  <div

                    key={expense.id}

                    className="flex items-center justify-between p-3 bg-gray-50 rounded-lg hover:bg-gray-100 transition-colors cursor-pointer"

                  >

                    <div className="flex-1 min-w-0">

                      <p className="text-sm font-medium text-gray-900 truncate">

                        {expense.description || expense.title || 'No description'}

                      </p>

                      <div className="flex items-center gap-2 mt-1">

                        <span className="text-xs text-gray-500 truncate">

                          {expense.category?.name || expense.category\_name || 'Uncategorized'}

                        </span>

                        <span className="text-xs text-gray-400">•</span>

                        <span className="text-xs text-gray-500 flex items-center gap-1">

                          <MdCalendarToday className="text-xs flex-shrink-0" />

                          {formatDate(expense.expense\_date || expense.date)}

                        </span>

                      </div>

                    </div>

                    <div className="text-right ml-2">

                      <p className="text-sm font-bold text-gray-900 whitespace-nowrap">

                        {formatCurrency(expense.amount)}

                      </p>

                    </div>

                  </div>

                ))

              ) : (

                <div className="text-center py-8 text-gray-400">

                  <MdAccountBalanceWallet className="text-5xl mx-auto mb-2 opacity-50" />

                  <p>No recent expenses</p>

                  <Link to="/expenses/add" className="text-blue-600 hover:text-blue-700 text-sm font-medium mt-2 inline-block">

                    Add your first expense

                  </Link>

                </div>

              )}

            </div>

          </div>

        </div>

        {/\* Category and Payment Breakdown \*/}

        <div className="grid grid-cols-1 lg:grid-cols-2 gap-6 mt-6">

          {/\* Category Breakdown \*/}

          <div className="bg-white rounded-xl shadow-lg p-6">

            <div className="flex items-center justify-between mb-6">

              <h2 className="text-xl font-bold text-gray-900">Top Categories</h2>

              <Link

                to="/analytics"

                className="text-blue-600 hover:text-blue-700 text-sm font-medium flex items-center gap-1"

              >

                View All <MdArrowForward />

              </Link>

            </div>

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

              {dashboardData.categoryBreakdown.length > 0 ? (

                dashboardData.categoryBreakdown.map((category, index) => (

                  <div key={index} className="space-y-2">

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

                      <div className="flex items-center gap-3 flex-1 min-w-0">

                        <div

                          className="w-3 h-3 rounded-full flex-shrink-0"

                          style={{ backgroundColor: category.color || '#3B82F6' }}

                        ></div>

                        <span className="text-sm font-medium text-gray-900 truncate">

                          {category.category}

                        </span>

                      </div>

                      <div className="text-right ml-2">

                        <p className="text-sm font-bold text-gray-900 whitespace-nowrap">

                          {formatCurrency(category.total)}

                        </p>

                        <p className="text-xs text-gray-500">

                          {(category.percentage || 0).toFixed(1)}%

                        </p>

                      </div>

                    </div>

                    <div className="w-full bg-gray-200 rounded-full h-2 overflow-hidden">

                      <div

                        className="h-2 rounded-full transition-all duration-300"

                        style={{

                          width: `${category.percentage || 0}%`,

                          backgroundColor: category.color || '#3B82F6',

                        }}

                      ></div>

                    </div>

                  </div>

                ))

              ) : (

                <div className="text-center py-8 text-gray-400">

                  <p>No category data available</p>

                </div>

              )}

            </div>

          </div>

          {/\* Payment Mode Breakdown \*/}

          <div className="bg-white rounded-xl shadow-lg p-6">

            <div className="flex items-center justify-between mb-6">

              <h2 className="text-xl font-bold text-gray-900">Payment Methods</h2>

              <Link

                to="/analytics"

                className="text-blue-600 hover:text-blue-700 text-sm font-medium flex items-center gap-1"

              >

                View All <MdArrowForward />

              </Link>

            </div>

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

              {dashboardData.paymentBreakdown.length > 0 ? (

                dashboardData.paymentBreakdown.map((payment, index) => (

                  <div key={index} className="space-y-2">

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

                      <span className="text-sm font-medium text-gray-900 truncate flex-1 min-w-0">

                        {payment.paymentmode || payment.payment\_mode}

                      </span>

                      <div className="text-right ml-2">

                        <p className="text-sm font-bold text-gray-900 whitespace-nowrap">

                          {formatCurrency(payment.total)}

                        </p>

                        <p className="text-xs text-gray-500">

                          {payment.count} transactions

                        </p>

                      </div>

                    </div>

                    <div className="w-full bg-gray-200 rounded-full h-2 overflow-hidden">

                      <div

                        className="bg-gradient-to-r from-purple-500 to-pink-500 h-2 rounded-full transition-all duration-300"

                        style={{ width: `${payment.percentage || 0}%` }}

                      ></div>

                    </div>

                  </div>

                ))

              ) : (

                <div className="text-center py-8 text-gray-400">

                  <p>No payment data available</p>

                </div>

              )}

            </div>

          </div>

        </div>

        {/\* Quick Actions \*/}

        <div className="mt-6 bg-gradient-to-r from-blue-500 to-purple-600 rounded-xl shadow-lg p-6">

          <div className="flex flex-col md:flex-row items-center justify-between gap-4">

            <div className="text-white text-center md:text-left">

              <h3 className="text-xl font-bold">Ready to track a new expense?</h3>

              <p className="text-blue-100 text-sm mt-1">

                Add your expenses quickly and keep your budget on track

              </p>

            </div>

            <Link

              to="/expenses/add"

              className="bg-white text-blue-600 px-6 py-3 rounded-lg font-semibold hover:bg-gray-100 transition-colors shadow-lg whitespace-nowrap"

            >

              Add Expense

            </Link>

          </div>

        </div>

      </div>

    </div>

  );

};

export default Dashboard;

E:\expense\_tracker\frontend\src\pages\expenses

E:\expense\_tracker\frontend\src\pages\expenses\AddExpense.jsx

/\*\*

 \* AddExpense Component

 \* Form for creating new expense

 \* Backend: POST /api/expenses

 \*/

import { useState, useEffect } from 'react';

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

import { toast } from 'react-hot-toast';

import { MdArrowBack, MdSave } from 'react-icons/md';

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

const AddExpense = () => {

  const navigate = useNavigate();

  const [formData, setFormData] = useState({

    category\_id: '',

    payment\_mode\_id: '',

    amount: '',

    description: '',

    expense\_date: new Date().toISOString().split('T')[0], // Today's date

  });

  const [categories, setCategories] = useState([]);

  const [paymentModes, setPaymentModes] = useState([]);

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

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

  // Fetch categories and payment modes

  useEffect(() => {

    const fetchMetadata = async () => {

      try {

        const [categoriesRes, paymentModesRes] = await Promise.all([

          expenseAPI.getCategories(),

          expenseAPI.getPaymentModes(),

        ]);

        setCategories(categoriesRes.data.categories || []);

        setPaymentModes(paymentModesRes.data.payment\_modes || []);

      } catch (error) {

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

        toast.error('Failed to load form data');

      }

    };

    fetchMetadata();

  }, []);

  const handleChange = (e) => {

    const { name, value } = e.target;

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

    // Clear error for this field

    if (errors[name]) {

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

    }

  };

  const validateForm = () => {

    const newErrors = {};

    // Category validation

    if (!formData.category\_id) {

      newErrors.category\_id = 'Category is required';

    }

    // Payment mode validation

    if (!formData.payment\_mode\_id) {

      newErrors.payment\_mode\_id = 'Payment mode is required';

    }

    // Amount validation - Backend requires positive number, max 99999999.99

    if (!formData.amount) {

      newErrors.amount = 'Amount is required';

    } else {

      const amount = parseFloat(formData.amount);

      if (isNaN(amount) || amount <= 0) {

        newErrors.amount = 'Amount must be greater than 0';

      } else if (amount > 99999999.99) {

        newErrors.amount = 'Amount is too large (max: 99,999,999.99)';

      }

    }

    // Description validation - Backend requires 3-255 characters

    if (!formData.description.trim()) {

      newErrors.description = 'Description is required';

    } else if (formData.description.trim().length < 3) {

      newErrors.description = 'Description must be at least 3 characters';

    } else if (formData.description.trim().length > 255) {

      newErrors.description = 'Description is too long (max: 255 characters)';

    }

    // Date validation - Backend checks date is not too old (5 years)

    if (!formData.expense\_date) {

      newErrors.expense\_date = 'Date is required';

    } else {

      const selectedDate = new Date(formData.expense\_date);

      const fiveYearsAgo = new Date();

      fiveYearsAgo.setFullYear(fiveYearsAgo.getFullYear() - 5);

      if (selectedDate < fiveYearsAgo) {

        newErrors.expense\_date = 'Date is too old (max 5 years ago)';

      } else if (selectedDate > new Date()) {

        newErrors.expense\_date = 'Date cannot be in the future';

      }

    }

    setErrors(newErrors);

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

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    if (!validateForm()) {

      toast.error('Please fix the errors in the form');

      return;

    }

    setLoading(true);

    try {

      // Backend expects: { category\_id, payment\_mode\_id, amount, description, expense\_date }

      const payload = {

        category\_id: parseInt(formData.category\_id),

        payment\_mode\_id: parseInt(formData.payment\_mode\_id),

        amount: parseFloat(formData.amount),

        description: formData.description.trim(),

        expense\_date: formData.expense\_date,

      };

      await expenseAPI.createExpense(payload);

      toast.success('Expense added successfully!');

      navigate('/expenses');

    } catch (error) {

      console.error('Error adding expense:', error);

      if (error.response?.status === 400) {

        toast.error(error.response?.data?.error || 'Invalid expense data');

      } else if (error.response?.status === 404) {

        toast.error('Invalid category or payment mode');

      } else {

        toast.error('Failed to add expense. Please try again.');

      }

    } finally {

      setLoading(false);

    }

  };

  return (

    <div className="min-h-screen bg-gray-50 p-6">

      <div className="max-w-3xl mx-auto">

        {/\* Header \*/}

        <div className="mb-6">

          <Link

            to="/expenses"

            className="inline-flex items-center gap-2 text-blue-600 hover:text-blue-700 mb-4"

          >

            <MdArrowBack className="text-xl" />

            Back to Expenses

          </Link>

          <h1 className="text-3xl font-bold text-gray-900">Add New Expense</h1>

          <p className="mt-1 text-sm text-gray-600">

            Fill in the details to track your expense

          </p>

        </div>

        {/\* Form \*/}

        <div className="bg-white rounded-xl shadow-lg p-6">

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

            {/\* Amount \*/}

            <div>

              <label htmlFor="amount" className="block text-sm font-medium text-gray-700 mb-2">

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

              </label>

              <div className="relative">

                <span className="absolute left-4 top-1/2 transform -translate-y-1/2 text-gray-500 font-medium">

                  ₹

                </span>

                <input

                  type="number"

                  id="amount"

                  name="amount"

                  value={formData.amount}

                  onChange={handleChange}

                  step="0.01"

                  min="0"

                  max="99999999.99"

                  className={`w-full pl-10 pr-4 py-3 border ${

                    errors.amount ? 'border-red-500' : 'border-gray-300'

                  } rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500 text-lg`}

                  placeholder="0.00"

                />

              </div>

              {errors.amount && (

                <p className="mt-1 text-sm text-red-500">{errors.amount}</p>

              )}

            </div>

            {/\* Description \*/}

            <div>

              <label htmlFor="description" className="block text-sm font-medium text-gray-700 mb-2">

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

              </label>

              <textarea

                id="description"

                name="description"

                value={formData.description}

                onChange={handleChange}

                rows="3"

                maxLength="255"

                className={`w-full px-4 py-3 border ${

                  errors.description ? 'border-red-500' : 'border-gray-300'

                } rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500 resize-none`}

                placeholder="Enter expense description (3-255 characters)"

              />

              <div className="flex justify-between mt-1">

                {errors.description ? (

                  <p className="text-sm text-red-500">{errors.description}</p>

                ) : (

                  <p className="text-sm text-gray-500">Minimum 3 characters</p>

                )}

                <p className="text-sm text-gray-500">

                  {formData.description.length}/255

                </p>

              </div>

            </div>

            {/\* Category \*/}

            <div>

              <label htmlFor="category\_id" className="block text-sm font-medium text-gray-700 mb-2">

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

              </label>

              <select

                id="category\_id"

                name="category\_id"

                value={formData.category\_id}

                onChange={handleChange}

                className={`w-full px-4 py-3 border ${

                  errors.category\_id ? 'border-red-500' : 'border-gray-300'

                } rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500`}

              >

                <option value="">Select a category</option>

                {categories.map((category) => (

                  <option key={category.id} value={category.id}>

                    {category.name}

                  </option>

                ))}

              </select>

              {errors.category\_id && (

                <p className="mt-1 text-sm text-red-500">{errors.category\_id}</p>

              )}

            </div>

            {/\* Payment Mode \*/}

            <div>

              <label htmlFor="payment\_mode\_id" className="block text-sm font-medium text-gray-700 mb-2">

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

              </label>

              <select

                id="payment\_mode\_id"

                name="payment\_mode\_id"

                value={formData.payment\_mode\_id}

                onChange={handleChange}

                className={`w-full px-4 py-3 border ${

                  errors.payment\_mode\_id ? 'border-red-500' : 'border-gray-300'

                } rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500`}

              >

                <option value="">Select a payment mode</option>

                {paymentModes.map((mode) => (

                  <option key={mode.id} value={mode.id}>

                    {mode.name}

                    {mode.bank\_name ? ` (${mode.bank\_name})` : ''}

                  </option>

                ))}

              </select>

              {errors.payment\_mode\_id && (

                <p className="mt-1 text-sm text-red-500">{errors.payment\_mode\_id}</p>

              )}

            </div>

            {/\* Expense Date \*/}

            <div>

              <label htmlFor="expense\_date" className="block text-sm font-medium text-gray-700 mb-2">

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

              </label>

              <input

                type="date"

                id="expense\_date"

                name="expense\_date"

                value={formData.expense\_date}

                onChange={handleChange}

                max={new Date().toISOString().split('T')[0]}

                className={`w-full px-4 py-3 border ${

                  errors.expense\_date ? 'border-red-500' : 'border-gray-300'

                } rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500`}

              />

              {errors.expense\_date && (

                <p className="mt-1 text-sm text-red-500">{errors.expense\_date}</p>

              )}

            </div>

            {/\* Submit Buttons \*/}

            <div className="flex gap-4 pt-4">

              <button

                type="button"

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

                className="flex-1 px-6 py-3 border border-gray-300 rounded-lg text-gray-700 font-medium hover:bg-gray-50 transition-colors"

              >

                Cancel

              </button>

              <button

                type="submit"

                disabled={loading}

                className="flex-1 flex items-center justify-center gap-2 px-6 py-3 bg-gradient-to-r from-blue-600 to-purple-600 text-white rounded-lg font-medium hover:from-blue-700 hover:to-purple-700 disabled:opacity-50 disabled:cursor-not-allowed transition-all"

              >

                {loading ? (

                  <>

                    <svg className="animate-spin h-5 w-5 text-white" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24">

                      <circle className="opacity-25" cx="12" cy="12" r="10" stroke="currentColor" strokeWidth="4"></circle>

                      <path className="opacity-75" fill="currentColor" d="M4 12a8 8 0 018-8V0C5.373 0 0 5.373 0 12h4zm2 5.291A7.962 7.962 0 014 12H0c0 3.042 1.135 5.824 3 7.938l3-2.647z"></path>

                    </svg>

                    Saving...

                  </>

                ) : (

                  <>

                    <MdSave className="text-xl" />

                    Save Expense

                  </>

                )}

              </button>

            </div>

          </form>

        </div>

      </div>

    </div>

  );

};

export default AddExpense;

E:\expense\_tracker\frontend\src\pages\expenses\EditExpense.jsx

/\*\*

 \* EditExpense Component

 \* Form for updating existing expense

 \* Backend: PUT /api/expenses/:id

 \*/

import { useState, useEffect } from 'react';

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

import { toast } from 'react-hot-toast';

import { MdArrowBack, MdSave } from 'react-icons/md';

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

const EditExpense = () => {

  const navigate = useNavigate();

  const { id } = useParams();

  const [formData, setFormData] = useState({

    category\_id: '',

    payment\_mode\_id: '',

    amount: '',

    description: '',

    expense\_date: '',

  });

  const [categories, setCategories] = useState([]);

  const [paymentModes, setPaymentModes] = useState([]);

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

  const [saving, setSaving] = useState(false);

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

  // Fetch expense details and metadata

  useEffect(() => {

    const fetchData = async () => {

      try {

        const [expenseRes, categoriesRes, paymentModesRes] = await Promise.all([

          expenseAPI.getExpenseById(id),

          expenseAPI.getCategories(),

          expenseAPI.getPaymentModes(),

        ]);

        const expense = expenseRes.data.expense;

        setFormData({

          category\_id: expense.category\_id,

          payment\_mode\_id: expense.payment\_mode\_id,

          amount: expense.amount,

          description: expense.description,

          expense\_date: expense.expense\_date,

        });

        setCategories(categoriesRes.data.categories || []);

        setPaymentModes(paymentModesRes.data.payment\_modes || []);

      } catch (error) {

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

        if (error.response?.status === 404) {

          toast.error('Expense not found');

          navigate('/expenses');

        } else {

          toast.error('Failed to load expense details');

        }

      } finally {

        setLoading(false);

      }

    };

    fetchData();

  }, [id, navigate]);

  const handleChange = (e) => {

    const { name, value } = e.target;

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

    if (errors[name]) {

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

    }

  };

  const validateForm = () => {

    const newErrors = {};

    if (!formData.category\_id) {

      newErrors.category\_id = 'Category is required';

    }

    if (!formData.payment\_mode\_id) {

      newErrors.payment\_mode\_id = 'Payment mode is required';

    }

    if (!formData.amount) {

      newErrors.amount = 'Amount is required';

    } else {

      const amount = parseFloat(formData.amount);

      if (isNaN(amount) || amount <= 0) {

        newErrors.amount = 'Amount must be greater than 0';

      } else if (amount > 99999999.99) {

        newErrors.amount = 'Amount is too large';

      }

    }

    if (!formData.description.trim()) {

      newErrors.description = 'Description is required';

    } else if (formData.description.trim().length < 3) {

      newErrors.description = 'Description must be at least 3 characters';

    } else if (formData.description.trim().length > 255) {

      newErrors.description = 'Description is too long';

    }

    if (!formData.expense\_date) {

      newErrors.expense\_date = 'Date is required';

    }

    setErrors(newErrors);

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

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    if (!validateForm()) {

      toast.error('Please fix the errors in the form');

      return;

    }

    setSaving(true);

    try {

      // Backend PUT endpoint accepts partial updates

      const payload = {

        category\_id: parseInt(formData.category\_id),

        payment\_mode\_id: parseInt(formData.payment\_mode\_id),

        amount: parseFloat(formData.amount),

        description: formData.description.trim(),

        expense\_date: formData.expense\_date,

      };

      await expenseAPI.updateExpense(id, payload);

      toast.success('Expense updated successfully!');

      navigate('/expenses');

    } catch (error) {

      console.error('Error updating expense:', error);

      if (error.response?.status === 404) {

        toast.error('Expense not found');

      } else if (error.response?.status === 400) {

        toast.error(error.response?.data?.error || 'Invalid expense data');

      } else {

        toast.error('Failed to update expense. Please try again.');

      }

    } finally {

      setSaving(false);

    }

  };

  if (loading) {

    return (

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

        <div className="text-center">

          <div className="animate-spin rounded-full h-16 w-16 border-b-2 border-blue-600 mx-auto"></div>

          <p className="mt-4 text-gray-600">Loading expense details...</p>

        </div>

      </div>

    );

  }

  return (

    <div className="min-h-screen bg-gray-50 p-6">

      <div className="max-w-3xl mx-auto">

        {/\* Header \*/}

        <div className="mb-6">

          <Link

            to="/expenses"

            className="inline-flex items-center gap-2 text-blue-600 hover:text-blue-700 mb-4"

          >

            <MdArrowBack className="text-xl" />

            Back to Expenses

          </Link>

          <h1 className="text-3xl font-bold text-gray-900">Edit Expense</h1>

          <p className="mt-1 text-sm text-gray-600">

            Update expense details

          </p>

        </div>

        {/\* Form \*/}

        <div className="bg-white rounded-xl shadow-lg p-6">

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

            {/\* Amount \*/}

            <div>

              <label htmlFor="amount" className="block text-sm font-medium text-gray-700 mb-2">

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

              </label>

              <div className="relative">

                <span className="absolute left-4 top-1/2 transform -translate-y-1/2 text-gray-500 font-medium">

                  ₹

                </span>

                <input

                  type="number"

                  id="amount"

                  name="amount"

                  value={formData.amount}

                  onChange={handleChange}

                  step="0.01"

                  min="0"

                  max="99999999.99"

                  className={`w-full pl-10 pr-4 py-3 border ${

                    errors.amount ? 'border-red-500' : 'border-gray-300'

                  } rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500 text-lg`}

                  placeholder="0.00"

                />

              </div>

              {errors.amount && (

                <p className="mt-1 text-sm text-red-500">{errors.amount}</p>

              )}

            </div>

            {/\* Description \*/}

            <div>

              <label htmlFor="description" className="block text-sm font-medium text-gray-700 mb-2">

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

              </label>

              <textarea

                id="description"

                name="description"

                value={formData.description}

                onChange={handleChange}

                rows="3"

                maxLength="255"

                className={`w-full px-4 py-3 border ${

                  errors.description ? 'border-red-500' : 'border-gray-300'

                } rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500 resize-none`}

                placeholder="Enter expense description"

              />

              <div className="flex justify-between mt-1">

                {errors.description ? (

                  <p className="text-sm text-red-500">{errors.description}</p>

                ) : (

                  <p className="text-sm text-gray-500">Minimum 3 characters</p>

                )}

                <p className="text-sm text-gray-500">

                  {formData.description.length}/255

                </p>

              </div>

            </div>

            {/\* Category \*/}

            <div>

              <label htmlFor="category\_id" className="block text-sm font-medium text-gray-700 mb-2">

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

              </label>

              <select

                id="category\_id"

                name="category\_id"

                value={formData.category\_id}

                onChange={handleChange}

                className={`w-full px-4 py-3 border ${

                  errors.category\_id ? 'border-red-500' : 'border-gray-300'

                } rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500`}

              >

                <option value="">Select a category</option>

                {categories.map((category) => (

                  <option key={category.id} value={category.id}>

                    {category.name}

                  </option>

                ))}

              </select>

              {errors.category\_id && (

                <p className="mt-1 text-sm text-red-500">{errors.category\_id}</p>

              )}

            </div>

            {/\* Payment Mode \*/}

            <div>

              <label htmlFor="payment\_mode\_id" className="block text-sm font-medium text-gray-700 mb-2">

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

              </label>

              <select

                id="payment\_mode\_id"

                name="payment\_mode\_id"

                value={formData.payment\_mode\_id}

                onChange={handleChange}

                className={`w-full px-4 py-3 border ${

                  errors.payment\_mode\_id ? 'border-red-500' : 'border-gray-300'

                } rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500`}

              >

                <option value="">Select a payment mode</option>

                {paymentModes.map((mode) => (

                  <option key={mode.id} value={mode.id}>

                    {mode.name}

                    {mode.bank\_name ? ` (${mode.bank\_name})` : ''}

                  </option>

                ))}

              </select>

              {errors.payment\_mode\_id && (

                <p className="mt-1 text-sm text-red-500">{errors.payment\_mode\_id}</p>

              )}

            </div>

            {/\* Expense Date \*/}

            <div>

              <label htmlFor="expense\_date" className="block text-sm font-medium text-gray-700 mb-2">

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

              </label>

              <input

                type="date"

                id="expense\_date"

                name="expense\_date"

                value={formData.expense\_date}

                onChange={handleChange}

                max={new Date().toISOString().split('T')[0]}

                className={`w-full px-4 py-3 border ${

                  errors.expense\_date ? 'border-red-500' : 'border-gray-300'

                } rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500`}

              />

              {errors.expense\_date && (

                <p className="mt-1 text-sm text-red-500">{errors.expense\_date}</p>

              )}

            </div>

            {/\* Submit Buttons \*/}

            <div className="flex gap-4 pt-4">

              <button

                type="button"

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

                className="flex-1 px-6 py-3 border border-gray-300 rounded-lg text-gray-700 font-medium hover:bg-gray-50 transition-colors"

              >

                Cancel

              </button>

              <button

                type="submit"

                disabled={saving}

                className="flex-1 flex items-center justify-center gap-2 px-6 py-3 bg-gradient-to-r from-blue-600 to-purple-600 text-white rounded-lg font-medium hover:from-blue-700 hover:to-purple-700 disabled:opacity-50 disabled:cursor-not-allowed transition-all"

              >

                {saving ? (

                  <>

                    <svg className="animate-spin h-5 w-5 text-white" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24">

                      <circle className="opacity-25" cx="12" cy="12" r="10" stroke="currentColor" strokeWidth="4"></circle>

                      <path className="opacity-75" fill="currentColor" d="M4 12a8 8 0 018-8V0C5.373 0 0 5.373 0 12h4zm2 5.291A7.962 7.962 0 014 12H0c0 3.042 1.135 5.824 3 7.938l3-2.647z"></path>

                    </svg>

                    Updating...

                  </>

                ) : (

                  <>

                    <MdSave className="text-xl" />

                    Update Expense

                  </>

                )}

              </button>

            </div>

          </form>

        </div>

      </div>

    </div>

  );

};

export default EditExpense;

E:\expense\_tracker\frontend\src\pages\expenses\ExpenseDetails.jsx

/\*\*

 \* ExpenseDetails Component

 \* View detailed information about a specific expense

 \* Backend: GET /api/expenses/:id

 \*/

import { useState, useEffect } from 'react';

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

import { toast } from 'react-hot-toast';

import {

  MdArrowBack,

  MdEdit,

  MdDelete,

  MdCalendarToday,

  MdCategory,

  MdPayment,

  MdDescription,

  MdAttachMoney,

} from 'react-icons/md';

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

import { formatCurrency, formatDate } from '../../utils/helpers';

const ExpenseDetails = () => {

  const { id } = useParams();

  const navigate = useNavigate();

  const [expense, setExpense] = useState(null);

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

  const [deleteModal, setDeleteModal] = useState(false);

  useEffect(() => {

    const fetchExpense = async () => {

      try {

        const response = await expenseAPI.getExpenseById(id);

        setExpense(response.data.expense);

      } catch (error) {

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

        if (error.response?.status === 404) {

          toast.error('Expense not found');

          navigate('/expenses');

        } else {

          toast.error('Failed to load expense details');

        }

      } finally {

        setLoading(false);

      }

    };

    fetchExpense();

  }, [id, navigate]);

  const handleDelete = async () => {

    try {

      await expenseAPI.deleteExpense(id);

      toast.success('Expense deleted successfully');

      navigate('/expenses');

    } catch (error) {

      console.error('Error deleting expense:', error);

      toast.error('Failed to delete expense');

    }

  };

  if (loading) {

    return (

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

        <div className="text-center">

          <div className="animate-spin rounded-full h-16 w-16 border-b-2 border-blue-600 mx-auto"></div>

          <p className="mt-4 text-gray-600">Loading expense details...</p>

        </div>

      </div>

    );

  }

  if (!expense) {

    return null;

  }

  return (

    <div className="min-h-screen bg-gray-50 p-6">

      <div className="max-w-4xl mx-auto">

        {/\* Header \*/}

        <div className="mb-6">

          <Link

            to="/expenses"

            className="inline-flex items-center gap-2 text-blue-600 hover:text-blue-700 mb-4"

          >

            <MdArrowBack className="text-xl" />

            Back to Expenses

          </Link>

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

            <h1 className="text-3xl font-bold text-gray-900">Expense Details</h1>

            <div className="flex gap-3">

              <Link

                to={`/expenses/edit/${id}`}

                className="flex items-center gap-2 px-4 py-2 bg-blue-600 text-white rounded-lg hover:bg-blue-700 transition-colors"

              >

                <MdEdit className="text-xl" />

                Edit

              </Link>

              <button

                onClick={() => setDeleteModal(true)}

                className="flex items-center gap-2 px-4 py-2 bg-red-600 text-white rounded-lg hover:bg-red-700 transition-colors"

              >

                <MdDelete className="text-xl" />

                Delete

              </button>

            </div>

          </div>

        </div>

        {/\* Main Content \*/}

        <div className="grid grid-cols-1 lg:grid-cols-3 gap-6">

          {/\* Amount Card \*/}

          <div className="lg:col-span-3 bg-gradient-to-br from-blue-500 to-purple-600 rounded-xl shadow-lg p-8 text-white">

            <div className="flex items-center gap-3 mb-2">

              <MdAttachMoney className="text-4xl" />

              <p className="text-blue-100 text-lg font-medium">Total Amount</p>

            </div>

            <h2 className="text-5xl font-bold">{formatCurrency(expense.amount)}</h2>

          </div>

          {/\* Details Card \*/}

          <div className="lg:col-span-2 bg-white rounded-xl shadow-lg p-6">

            <h3 className="text-xl font-bold text-gray-900 mb-6">Expense Information</h3>

            <div className="space-y-6">

              {/\* Description \*/}

              <div className="flex gap-4">

                <div className="flex-shrink-0">

                  <div className="w-12 h-12 bg-gray-100 rounded-lg flex items-center justify-center">

                    <MdDescription className="text-2xl text-gray-600" />

                  </div>

                </div>

                <div className="flex-1">

                  <p className="text-sm font-medium text-gray-500 mb-1">Description</p>

                  <p className="text-lg text-gray-900">{expense.description}</p>

                </div>

              </div>

              {/\* Category \*/}

              <div className="flex gap-4">

                <div className="flex-shrink-0">

                  <div

                    className="w-12 h-12 rounded-lg flex items-center justify-center"

                    style={{

                      backgroundColor: expense.category?.color + '20' || '#3B82F620',

                    }}

                  >

                    <MdCategory

                      className="text-2xl"

                      style={{ color: expense.category?.color || '#3B82F6' }}

                    />

                  </div>

                </div>

                <div className="flex-1">

                  <p className="text-sm font-medium text-gray-500 mb-1">Category</p>

                  <p className="text-lg text-gray-900">{expense.category?.name}</p>

                  {expense.category?.description && (

                    <p className="text-sm text-gray-500 mt-1">

                      {expense.category.description}

                    </p>

                  )}

                </div>

              </div>

              {/\* Payment Mode \*/}

              <div className="flex gap-4">

                <div className="flex-shrink-0">

                  <div className="w-12 h-12 bg-green-100 rounded-lg flex items-center justify-center">

                    <MdPayment className="text-2xl text-green-600" />

                  </div>

                </div>

                <div className="flex-1">

                  <p className="text-sm font-medium text-gray-500 mb-1">Payment Mode</p>

                  <p className="text-lg text-gray-900">

                    {expense.payment\_mode?.name}

                    {expense.payment\_mode?.bank\_name && (

                      <span className="text-gray-600 text-base ml-2">

                        ({expense.payment\_mode.bank\_name})

                      </span>

                    )}

                  </p>

                </div>

              </div>

              {/\* Date \*/}

              <div className="flex gap-4">

                <div className="flex-shrink-0">

                  <div className="w-12 h-12 bg-blue-100 rounded-lg flex items-center justify-center">

                    <MdCalendarToday className="text-2xl text-blue-600" />

                  </div>

                </div>

                <div className="flex-1">

                  <p className="text-sm font-medium text-gray-500 mb-1">Expense Date</p>

                  <p className="text-lg text-gray-900">{formatDate(expense.expense\_date)}</p>

                </div>

              </div>

            </div>

          </div>

          {/\* Metadata Card \*/}

          <div className="bg-white rounded-xl shadow-lg p-6">

            <h3 className="text-xl font-bold text-gray-900 mb-6">Metadata</h3>

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

              <div>

                <p className="text-sm font-medium text-gray-500 mb-1">Created At</p>

                <p className="text-sm text-gray-900">

                  {new Date(expense.created\_at).toLocaleString()}

                </p>

              </div>

              <div>

                <p className="text-sm font-medium text-gray-500 mb-1">Last Updated</p>

                <p className="text-sm text-gray-900">

                  {new Date(expense.updated\_at).toLocaleString()}

                </p>

              </div>

              <div>

                <p className="text-sm font-medium text-gray-500 mb-1">Expense ID</p>

                <p className="text-sm text-gray-900 font-mono">#{expense.id}</p>

              </div>

            </div>

          </div>

        </div>

      </div>

      {/\* Delete Confirmation Modal \*/}

      {deleteModal && (

        <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center z-50 p-4">

          <div className="bg-white rounded-xl shadow-2xl max-w-md w-full p-6">

            <h3 className="text-xl font-bold text-gray-900 mb-2">Delete Expense</h3>

            <p className="text-gray-600 mb-6">

              Are you sure you want to delete this expense? This action cannot be undone.

            </p>

            <div className="bg-red-50 rounded-lg p-4 mb-6">

              <p className="text-sm text-gray-900 font-medium">

                {expense.description}

              </p>

              <p className="text-2xl font-bold text-red-600 mt-2">

                {formatCurrency(expense.amount)}

              </p>

            </div>

            <div className="flex gap-3">

              <button

                onClick={() => setDeleteModal(false)}

                className="flex-1 px-4 py-2 border border-gray-300 rounded-lg text-gray-700 font-medium hover:bg-gray-50 transition-colors"

              >

                Cancel

              </button>

              <button

                onClick={handleDelete}

                className="flex-1 px-4 py-2 bg-red-600 text-white rounded-lg font-medium hover:bg-red-700 transition-colors"

              >

                Delete

              </button>

            </div>

          </div>

        </div>

      )}

    </div>

  );

};

export default ExpenseDetails;

E:\expense\_tracker\frontend\src\pages\expenses\ExpenseList.jsx

/\*\*

 \* ExpenseList Component

 \* Displays paginated list of expenses with filters

 \* Backend: GET /api/expenses

 \*/

import { useState, useEffect } from 'react';

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

import { toast } from 'react-hot-toast';

import {

  MdAdd,

  MdEdit,

  MdDelete,

  MdFilterList,

  MdSearch,

  MdCalendarToday,

  MdAttachMoney,

} from 'react-icons/md';

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

import { formatCurrency, formatDate } from '../../utils/helpers';

const ExpenseList = () => {

  const navigate = useNavigate();

  const [expenses, setExpenses] = useState([]);

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

  const [categories, setCategories] = useState([]);

  const [paymentModes, setPaymentModes] = useState([]);

  // Pagination

  const [pagination, setPagination] = useState({

    total: 0,

    limit: 20,

    offset: 0,

    currentPage: 1,

  });

  // Filters

  const [filters, setFilters] = useState({

    startDate: '',

    endDate: '',

    categoryId: '',

    paymentModeId: '',

    searchTerm: '',

  });

  const [showFilters, setShowFilters] = useState(false);

  const [deleteModal, setDeleteModal] = useState({ show: false, expense: null });

  // Fetch expenses with filters

  const fetchExpenses = async () => {

    try {

      setLoading(true);

      const params = {

        limit: pagination.limit,

        offset: pagination.offset,

      };

      if (filters.startDate) params.start\_date = filters.startDate;

      if (filters.endDate) params.end\_date = filters.endDate;

      if (filters.categoryId) params.category\_id = filters.categoryId;

      if (filters.paymentModeId) params.payment\_mode\_id = filters.paymentModeId;

      const response = await expenseAPI.getExpenses(params);

      // Backend returns: { expenses: [], total\_count: 0, limit: 20, offset: 0 }

      setExpenses(response.data.expenses || []);

      setPagination((prev) => ({

        ...prev,

        total: response.data.total\_count || 0,

      }));

    } catch (error) {

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

      toast.error('Failed to load expenses');

    } finally {

      setLoading(false);

    }

  };

  // Fetch categories and payment modes

  const fetchMetadata = async () => {

    try {

      const [categoriesRes, paymentModesRes] = await Promise.all([

        expenseAPI.getCategories(),

        expenseAPI.getPaymentModes(),

      ]);

      setCategories(categoriesRes.data.categories || []);

      setPaymentModes(paymentModesRes.data.payment\_modes || []);

    } catch (error) {

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

    }

  };

  useEffect(() => {

    fetchMetadata();

  }, []);

  useEffect(() => {

    fetchExpenses();

  }, [pagination.offset, filters]);

  const handleFilterChange = (key, value) => {

    setFilters((prev) => ({ ...prev, [key]: value }));

    setPagination((prev) => ({ ...prev, offset: 0, currentPage: 1 }));

  };

  const handleClearFilters = () => {

    setFilters({

      startDate: '',

      endDate: '',

      categoryId: '',

      paymentModeId: '',

      searchTerm: '',

    });

  };

  const handlePageChange = (newPage) => {

    const newOffset = (newPage - 1) \* pagination.limit;

    setPagination((prev) => ({

      ...prev,

      offset: newOffset,

      currentPage: newPage,

    }));

  };

  const handleDelete = async () => {

    if (!deleteModal.expense) return;

    try {

      await expenseAPI.deleteExpense(deleteModal.expense.id);

      toast.success('Expense deleted successfully');

      setDeleteModal({ show: false, expense: null });

      fetchExpenses();

    } catch (error) {

      console.error('Error deleting expense:', error);

      toast.error('Failed to delete expense');

    }

  };

  const totalPages = Math.ceil(pagination.total / pagination.limit);

  // Filter expenses by search term (client-side)

  const filteredExpenses = expenses.filter((expense) =>

    filters.searchTerm

      ? expense.description?.toLowerCase().includes(filters.searchTerm.toLowerCase())

      : true

  );

  if (loading && expenses.length === 0) {

    return (

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

        <div className="text-center">

          <div className="animate-spin rounded-full h-16 w-16 border-b-2 border-blue-600 mx-auto"></div>

          <p className="mt-4 text-gray-600">Loading expenses...</p>

        </div>

      </div>

    );

  }

  return (

    <div className="min-h-screen bg-gray-50 p-6">

      <div className="max-w-7xl mx-auto">

        {/\* Header \*/}

        <div className="flex flex-col md:flex-row md:items-center md:justify-between mb-6">

          <div>

            <h1 className="text-3xl font-bold text-gray-900">Expenses</h1>

            <p className="mt-1 text-sm text-gray-600">

              Manage and track your expenses

            </p>

          </div>

          <div className="mt-4 md:mt-0 flex gap-3">

            <button

              onClick={() => setShowFilters(!showFilters)}

              className="flex items-center gap-2 px-4 py-2 bg-white rounded-lg shadow-sm text-gray-700 hover:bg-gray-50 transition-colors"

            >

              <MdFilterList className="text-xl" />

              Filters

            </button>

            <Link

              to="/expenses/add"

              className="flex items-center gap-2 px-4 py-2 bg-gradient-to-r from-blue-600 to-purple-600 text-white rounded-lg shadow-lg hover:from-blue-700 hover:to-purple-700 transition-all"

            >

              <MdAdd className="text-xl" />

              Add Expense

            </Link>

          </div>

        </div>

        {/\* Filters Panel \*/}

        {showFilters && (

          <div className="bg-white rounded-xl shadow-lg p-6 mb-6">

            <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-4">

              {/\* Start Date \*/}

              <div>

                <label className="block text-sm font-medium text-gray-700 mb-2">

                  Start Date

                </label>

                <input

                  type="date"

                  value={filters.startDate}

                  onChange={(e) => handleFilterChange('startDate', e.target.value)}

                  className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500"

                />

              </div>

              {/\* End Date \*/}

              <div>

                <label className="block text-sm font-medium text-gray-700 mb-2">

                  End Date

                </label>

                <input

                  type="date"

                  value={filters.endDate}

                  onChange={(e) => handleFilterChange('endDate', e.target.value)}

                  className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500"

                />

              </div>

              {/\* Category Filter \*/}

              <div>

                <label className="block text-sm font-medium text-gray-700 mb-2">

                  Category

                </label>

                <select

                  value={filters.categoryId}

                  onChange={(e) => handleFilterChange('categoryId', e.target.value)}

                  className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500"

                >

                  <option value="">All Categories</option>

                  {categories.map((cat) => (

                    <option key={cat.id} value={cat.id}>

                      {cat.name}

                    </option>

                  ))}

                </select>

              </div>

              {/\* Payment Mode Filter \*/}

              <div>

                <label className="block text-sm font-medium text-gray-700 mb-2">

                  Payment Mode

                </label>

                <select

                  value={filters.paymentModeId}

                  onChange={(e) => handleFilterChange('paymentModeId', e.target.value)}

                  className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500"

                >

                  <option value="">All Payment Modes</option>

                  {paymentModes.map((pm) => (

                    <option key={pm.id} value={pm.id}>

                      {pm.name} {pm.bank\_name ? `(${pm.bank\_name})` : ''}

                    </option>

                  ))}

                </select>

              </div>

            </div>

            {/\* Clear Filters Button \*/}

            <div className="mt-4 flex justify-end">

              <button

                onClick={handleClearFilters}

                className="px-4 py-2 text-sm text-gray-600 hover:text-gray-800 font-medium"

              >

                Clear All Filters

              </button>

            </div>

          </div>

        )}

        {/\* Search Bar \*/}

        <div className="bg-white rounded-xl shadow-lg p-4 mb-6">

          <div className="relative">

            <MdSearch className="absolute left-4 top-1/2 transform -translate-y-1/2 text-gray-400 text-xl" />

            <input

              type="text"

              placeholder="Search expenses by description..."

              value={filters.searchTerm}

              onChange={(e) => handleFilterChange('searchTerm', e.target.value)}

              className="w-full pl-12 pr-4 py-3 border border-gray-300 rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500"

            />

          </div>

        </div>

        {/\* Expenses List \*/}

        <div className="bg-white rounded-xl shadow-lg overflow-hidden">

          {filteredExpenses.length > 0 ? (

            <>

              {/\* Desktop Table View \*/}

              <div className="hidden md:block overflow-x-auto">

                <table className="w-full">

                  <thead className="bg-gray-50 border-b border-gray-200">

                    <tr>

                      <th className="px-6 py-4 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">

                        Date

                      </th>

                      <th className="px-6 py-4 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">

                        Description

                      </th>

                      <th className="px-6 py-4 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">

                        Category

                      </th>

                      <th className="px-6 py-4 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">

                        Payment Mode

                      </th>

                      <th className="px-6 py-4 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">

                        Amount

                      </th>

                      <th className="px-6 py-4 text-right text-xs font-medium text-gray-500 uppercase tracking-wider">

                        Actions

                      </th>

                    </tr>

                  </thead>

                  <tbody className="bg-white divide-y divide-gray-200">

                    {filteredExpenses.map((expense) => (

                      <tr

                        key={expense.id}

                        className="hover:bg-gray-50 transition-colors cursor-pointer"

                        onClick={() => navigate(`/expenses/${expense.id}`)}

                      >

                        <td className="px-6 py-4 whitespace-nowrap text-sm text-gray-900">

                          <div className="flex items-center gap-2">

                            <MdCalendarToday className="text-gray-400" />

                            {formatDate(expense.expense\_date)}

                          </div>

                        </td>

                        <td className="px-6 py-4 text-sm text-gray-900">

                          {expense.description}

                        </td>

                        <td className="px-6 py-4 whitespace-nowrap text-sm">

                          <span

                            className="px-3 py-1 rounded-full text-xs font-medium"

                            style={{

                              backgroundColor: expense.category?.color + '20' || '#3B82F620',

                              color: expense.category?.color || '#3B82F6',

                            }}

                          >

                            {expense.category?.name}

                          </span>

                        </td>

                        <td className="px-6 py-4 whitespace-nowrap text-sm text-gray-900">

                          {expense.payment\_mode?.name}

                          {expense.payment\_mode?.bank\_name && (

                            <span className="text-gray-500 text-xs ml-1">

                              ({expense.payment\_mode.bank\_name})

                            </span>

                          )}

                        </td>

                        <td className="px-6 py-4 whitespace-nowrap text-sm font-semibold text-gray-900">

                          {formatCurrency(expense.amount)}

                        </td>

                        <td className="px-6 py-4 whitespace-nowrap text-right text-sm font-medium">

                          <div className="flex items-center justify-end gap-2">

                            <button

                              onClick={(e) => {

                                e.stopPropagation();

                                navigate(`/expenses/edit/${expense.id}`);

                              }}

                              className="text-blue-600 hover:text-blue-900 p-2"

                            >

                              <MdEdit className="text-xl" />

                            </button>

                            <button

                              onClick={(e) => {

                                e.stopPropagation();

                                setDeleteModal({ show: true, expense });

                              }}

                              className="text-red-600 hover:text-red-900 p-2"

                            >

                              <MdDelete className="text-xl" />

                            </button>

                          </div>

                        </td>

                      </tr>

                    ))}

                  </tbody>

                </table>

              </div>

              {/\* Mobile Card View \*/}

              <div className="md:hidden divide-y divide-gray-200">

                {filteredExpenses.map((expense) => (

                  <div

                    key={expense.id}

                    className="p-4 hover:bg-gray-50 transition-colors"

                    onClick={() => navigate(`/expenses/${expense.id}`)}

                  >

                    <div className="flex justify-between items-start mb-2">

                      <div>

                        <p className="font-medium text-gray-900">

                          {expense.description}

                        </p>

                        <p className="text-xs text-gray-500 mt-1 flex items-center gap-1">

                          <MdCalendarToday />

                          {formatDate(expense.expense\_date)}

                        </p>

                      </div>

                      <p className="text-lg font-bold text-gray-900">

                        {formatCurrency(expense.amount)}

                      </p>

                    </div>

                    <div className="flex items-center gap-2 mt-2">

                      <span

                        className="px-3 py-1 rounded-full text-xs font-medium"

                        style={{

                          backgroundColor: expense.category?.color + '20' || '#3B82F620',

                          color: expense.category?.color || '#3B82F6',

                        }}

                      >

                        {expense.category?.name}

                      </span>

                      <span className="text-xs text-gray-600">

                        {expense.payment\_mode?.name}

                      </span>

                    </div>

                    <div className="flex gap-2 mt-3">

                      <button

                        onClick={(e) => {

                          e.stopPropagation();

                          navigate(`/expenses/edit/${expense.id}`);

                        }}

                        className="flex-1 flex items-center justify-center gap-1 px-3 py-2 bg-blue-50 text-blue-600 rounded-lg text-sm font-medium"

                      >

                        <MdEdit /> Edit

                      </button>

                      <button

                        onClick={(e) => {

                          e.stopPropagation();

                          setDeleteModal({ show: true, expense });

                        }}

                        className="flex-1 flex items-center justify-center gap-1 px-3 py-2 bg-red-50 text-red-600 rounded-lg text-sm font-medium"

                      >

                        <MdDelete /> Delete

                      </button>

                    </div>

                  </div>

                ))}

              </div>

              {/\* Pagination \*/}

              {totalPages > 1 && (

                <div className="bg-gray-50 px-6 py-4 flex items-center justify-between border-t border-gray-200">

                  <div className="text-sm text-gray-700">

                    Showing {pagination.offset + 1} to{' '}

                    {Math.min(pagination.offset + pagination.limit, pagination.total)} of{' '}

                    {pagination.total} expenses

                  </div>

                  <div className="flex gap-2">

                    <button

                      onClick={() => handlePageChange(pagination.currentPage - 1)}

                      disabled={pagination.currentPage === 1}

                      className="px-4 py-2 border border-gray-300 rounded-lg text-sm font-medium text-gray-700 bg-white hover:bg-gray-50 disabled:opacity-50 disabled:cursor-not-allowed"

                    >

                      Previous

                    </button>

                    {[...Array(totalPages)].map((\_, index) => {

                      const page = index + 1;

                      if (

                        page === 1 ||

                        page === totalPages ||

                        (page >= pagination.currentPage - 1 && page <= pagination.currentPage + 1)

                      ) {

                        return (

                          <button

                            key={page}

                            onClick={() => handlePageChange(page)}

                            className={`px-4 py-2 border rounded-lg text-sm font-medium ${

                              page === pagination.currentPage

                                ? 'bg-blue-600 text-white border-blue-600'

                                : 'bg-white text-gray-700 border-gray-300 hover:bg-gray-50'

                            }`}

                          >

                            {page}

                          </button>

                        );

                      } else if (

                        page === pagination.currentPage - 2 ||

                        page === pagination.currentPage + 2

                      ) {

                        return <span key={page} className="px-2">...</span>;

                      }

                      return null;

                    })}

                    <button

                      onClick={() => handlePageChange(pagination.currentPage + 1)}

                      disabled={pagination.currentPage === totalPages}

                      className="px-4 py-2 border border-gray-300 rounded-lg text-sm font-medium text-gray-700 bg-white hover:bg-gray-50 disabled:opacity-50 disabled:cursor-not-allowed"

                    >

                      Next

                    </button>

                  </div>

                </div>

              )}

            </>

          ) : (

            <div className="text-center py-12">

              <MdAttachMoney className="mx-auto text-6xl text-gray-300 mb-4" />

              <p className="text-gray-500 text-lg mb-4">No expenses found</p>

              <Link

                to="/expenses/add"

                className="inline-flex items-center gap-2 px-6 py-3 bg-blue-600 text-white rounded-lg hover:bg-blue-700 transition-colors"

              >

                <MdAdd className="text-xl" />

                Add Your First Expense

              </Link>

            </div>

          )}

        </div>

      </div>

      {/\* Delete Confirmation Modal \*/}

      {deleteModal.show && (

        <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center z-50 p-4">

          <div className="bg-white rounded-xl shadow-2xl max-w-md w-full p-6">

            <h3 className="text-xl font-bold text-gray-900 mb-2">Delete Expense</h3>

            <p className="text-gray-600 mb-6">

              Are you sure you want to delete this expense? This action cannot be undone.

            </p>

            <div className="bg-gray-50 rounded-lg p-4 mb-6">

              <p className="text-sm text-gray-600">

                <strong>Description:</strong> {deleteModal.expense?.description}

              </p>

              <p className="text-sm text-gray-600 mt-1">

                <strong>Amount:</strong> {formatCurrency(deleteModal.expense?.amount)}

              </p>

            </div>

            <div className="flex gap-3">

              <button

                onClick={() => setDeleteModal({ show: false, expense: null })}

                className="flex-1 px-4 py-2 border border-gray-300 rounded-lg text-gray-700 font-medium hover:bg-gray-50 transition-colors"

              >

                Cancel

              </button>

              <button

                onClick={handleDelete}

                className="flex-1 px-4 py-2 bg-red-600 text-white rounded-lg font-medium hover:bg-red-700 transition-colors"

              >

                Delete

              </button>

            </div>

          </div>

        </div>

      )}

    </div>

  );

};

export default ExpenseList;

E:\expense\_tracker\frontend\src\pages\export

E:\expense\_tracker\frontend\src\pages\export\ExportReports.jsx

/\*\*

 \* ExportReports Component

 \* Export expenses to CSV and PDF formats with filters

 \* Backend: GET /api/export/csv, GET /api/export/pdf

 \*/

import { useState, useEffect } from 'react';

import { toast } from 'react-hot-toast';

import {

  MdFileDownload,

  MdPictureAsPdf,

  MdTableChart,

  MdFilterList,

  MdCalendarToday,

  MdCheckCircle,

} from 'react-icons/md';

import { exportAPI, expenseAPI } from '../../services/api';

import { formatDate } from '../../utils/helpers';

const ExportReports = () => {

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

  const [categories, setCategories] = useState([]);

  const [paymentModes, setPaymentModes] = useState([]);

  // Export filters

  const [exportConfig, setExportConfig] = useState({

    format: 'pdf', // 'pdf' or 'csv'

    reportType: 'detailed', // 'detailed' or 'summary' (PDF only)

    period: 'month', // 'today', 'week', 'month', 'year', 'custom'

    startDate: '',

    endDate: '',

    categoryId: '',

    paymentModeId: '',

  });

  const [exportHistory, setExportHistory] = useState([]);

  // Period options

  const periodOptions = [

    { value: 'today', label: 'Today' },

    { value: 'week', label: 'This Week' },

    { value: 'month', label: 'This Month' },

    { value: 'year', label: 'This Year' },

    { value: 'custom', label: 'Custom Date Range' },

  ];

  // Fetch categories and payment modes

  useEffect(() => {

    const fetchMetadata = async () => {

      try {

        const [categoriesRes, paymentModesRes] = await Promise.all([

          expenseAPI.getCategories(),

          expenseAPI.getPaymentModes(),

        ]);

        setCategories(categoriesRes.data.categories || []);

        setPaymentModes(paymentModesRes.data.payment\_modes || []);

      } catch (error) {

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

      }

    };

    fetchMetadata();

    // Load export history from localStorage

    const history = JSON.parse(localStorage.getItem('exportHistory') || '[]');

    setExportHistory(history);

  }, []);

  const handleConfigChange = (key, value) => {

    setExportConfig((prev) => ({ ...prev, [key]: value }));

  };

  const addToHistory = (exportRecord) => {

    const newHistory = [exportRecord, ...exportHistory].slice(0, 10); // Keep last 10

    setExportHistory(newHistory);

    localStorage.setItem('exportHistory', JSON.stringify(newHistory));

  };

  const handleExport = async () => {

    try {

      setLoading(true);

      // Build query parameters

      const params = new URLSearchParams();

      // Add period or custom dates

      if (exportConfig.period === 'custom') {

        if (!exportConfig.startDate || !exportConfig.endDate) {

          toast.error('Please select both start and end dates for custom range');

          setLoading(false);

          return;

        }

        params.append('start\_date', exportConfig.startDate);

        params.append('end\_date', exportConfig.endDate);

      } else {

        params.append('period', exportConfig.period);

      }

      // Add optional filters

      if (exportConfig.categoryId) {

        params.append('category\_id', exportConfig.categoryId);

      }

      if (exportConfig.paymentModeId) {

        params.append('payment\_mode\_id', exportConfig.paymentModeId);

      }

      // Add report type for PDF

      if (exportConfig.format === 'pdf') {

        params.append('report\_type', exportConfig.reportType);

      }

      // Make export request

      let response;

      if (exportConfig.format === 'csv') {

        response = await exportAPI.exportCSV(params.toString());

      } else {

        response = await exportAPI.exportPDF(params.toString());

      }

      // Create blob and download

      const blob = new Blob([response.data], {

        type:

          exportConfig.format === 'csv'

            ? 'text/csv'

            : 'application/pdf',

      });

      const url = window.URL.createObjectURL(blob);

      const link = document.createElement('a');

      link.href = url;

      // Generate filename

      const timestamp = new Date().toISOString().split('T')[0];

      const filename =

        exportConfig.format === 'csv'

          ? `expenses\_${timestamp}.csv`

          : `expense\_report\_${timestamp}.pdf`;

      link.setAttribute('download', filename);

      document.body.appendChild(link);

      link.click();

      link.remove();

      window.URL.revokeObjectURL(url);

      // Add to export history

      const exportRecord = {

        id: Date.now(),

        format: exportConfig.format.toUpperCase(),

        reportType: exportConfig.reportType,

        period: exportConfig.period,

        timestamp: new Date().toISOString(),

        filename,

      };

      addToHistory(exportRecord);

      toast.success(`${exportConfig.format.toUpperCase()} exported successfully!`);

    } catch (error) {

      console.error('Export error:', error);

      toast.error(

        error.response?.data?.error || 'Failed to export. Please try again.'

      );

    } finally {

      setLoading(false);

    }

  };

  const getPeriodLabel = (period) => {

    const option = periodOptions.find((opt) => opt.value === period);

    return option ? option.label : period;

  };

  return (

    <div className="min-h-screen bg-gray-50 p-6">

      <div className="max-w-7xl mx-auto">

        {/\* Header \*/}

        <div className="mb-8">

          <div className="flex items-center gap-3 mb-2">

            <div className="w-12 h-12 bg-gradient-to-br from-green-500 to-teal-500 rounded-xl flex items-center justify-center">

              <MdFileDownload className="text-3xl text-white" />

            </div>

            <div>

              <h1 className="text-3xl font-bold text-gray-900">Export Reports</h1>

              <p className="text-sm text-gray-600">

                Download your expense data in CSV or PDF format

              </p>

            </div>

          </div>

        </div>

        <div className="grid grid-cols-1 lg:grid-cols-3 gap-6">

          {/\* Export Configuration \*/}

          <div className="lg:col-span-2 space-y-6">

            {/\* Format Selection \*/}

            <div className="bg-white rounded-xl shadow-lg p-6">

              <h2 className="text-xl font-bold text-gray-900 mb-6 flex items-center gap-2">

                <MdFilterList className="text-2xl text-blue-600" />

                Export Configuration

              </h2>

              {/\* Format Type \*/}

              <div className="mb-6">

                <label className="block text-sm font-medium text-gray-700 mb-3">

                  Export Format

                </label>

                <div className="grid grid-cols-2 gap-4">

                  {/\* CSV Option \*/}

                  <button

                    onClick={() => handleConfigChange('format', 'csv')}

                    className={`p-6 rounded-xl border-2 transition-all ${

                      exportConfig.format === 'csv'

                        ? 'border-green-500 bg-green-50'

                        : 'border-gray-200 hover:border-green-300'

                    }`}

                  >

                    <div className="flex flex-col items-center gap-3">

                      <MdTableChart

                        className={`text-4xl ${

                          exportConfig.format === 'csv'

                            ? 'text-green-600'

                            : 'text-gray-400'

                        }`}

                      />

                      <div className="text-center">

                        <p className="font-bold text-gray-900">CSV</p>

                        <p className="text-xs text-gray-500 mt-1">

                          Spreadsheet format

                        </p>

                      </div>

                      {exportConfig.format === 'csv' && (

                        <MdCheckCircle className="text-2xl text-green-600" />

                      )}

                    </div>

                  </button>

                  {/\* PDF Option \*/}

                  <button

                    onClick={() => handleConfigChange('format', 'pdf')}

                    className={`p-6 rounded-xl border-2 transition-all ${

                      exportConfig.format === 'pdf'

                        ? 'border-red-500 bg-red-50'

                        : 'border-gray-200 hover:border-red-300'

                    }`}

                  >

                    <div className="flex flex-col items-center gap-3">

                      <MdPictureAsPdf

                        className={`text-4xl ${

                          exportConfig.format === 'pdf'

                            ? 'text-red-600'

                            : 'text-gray-400'

                        }`}

                      />

                      <div className="text-center">

                        <p className="font-bold text-gray-900">PDF</p>

                        <p className="text-xs text-gray-500 mt-1">

                          Professional report

                        </p>

                      </div>

                      {exportConfig.format === 'pdf' && (

                        <MdCheckCircle className="text-2xl text-red-600" />

                      )}

                    </div>

                  </button>

                </div>

              </div>

              {/\* PDF Report Type \*/}

              {exportConfig.format === 'pdf' && (

                <div className="mb-6 p-4 bg-red-50 rounded-lg border border-red-200">

                  <label className="block text-sm font-medium text-gray-700 mb-3">

                    PDF Report Type

                  </label>

                  <div className="grid grid-cols-2 gap-3">

                    <button

                      onClick={() => handleConfigChange('reportType', 'summary')}

                      className={`px-4 py-3 rounded-lg font-medium transition-colors ${

                        exportConfig.reportType === 'summary'

                          ? 'bg-red-600 text-white'

                          : 'bg-white text-gray-700 hover:bg-gray-100'

                      }`}

                    >

                      Summary

                    </button>

                    <button

                      onClick={() => handleConfigChange('reportType', 'detailed')}

                      className={`px-4 py-3 rounded-lg font-medium transition-colors ${

                        exportConfig.reportType === 'detailed'

                          ? 'bg-red-600 text-white'

                          : 'bg-white text-gray-700 hover:bg-gray-100'

                      }`}

                    >

                      Detailed

                    </button>

                  </div>

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

                    {exportConfig.reportType === 'summary'

                      ? 'Summary report includes totals and category breakdown only'

                      : 'Detailed report includes all individual expense transactions'}

                  </p>

                </div>

              )}

              {/\* Time Period \*/}

              <div className="mb-6">

                <label className="block text-sm font-medium text-gray-700 mb-3">

                  Time Period

                </label>

                <div className="grid grid-cols-2 md:grid-cols-3 gap-3">

                  {periodOptions.map((option) => (

                    <button

                      key={option.value}

                      onClick={() => handleConfigChange('period', option.value)}

                      className={`px-4 py-2 rounded-lg font-medium transition-colors ${

                        exportConfig.period === option.value

                          ? 'bg-blue-600 text-white'

                          : 'bg-gray-100 text-gray-700 hover:bg-gray-200'

                      }`}

                    >

                      {option.label}

                    </button>

                  ))}

                </div>

              </div>

              {/\* Custom Date Range \*/}

              {exportConfig.period === 'custom' && (

                <div className="mb-6 p-4 bg-blue-50 rounded-lg border border-blue-200">

                  <div className="grid grid-cols-1 md:grid-cols-2 gap-4">

                    <div>

                      <label className="block text-sm font-medium text-gray-700 mb-2">

                        Start Date

                      </label>

                      <input

                        type="date"

                        value={exportConfig.startDate}

                        onChange={(e) =>

                          handleConfigChange('startDate', e.target.value)

                        }

                        className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500"

                      />

                    </div>

                    <div>

                      <label className="block text-sm font-medium text-gray-700 mb-2">

                        End Date

                      </label>

                      <input

                        type="date"

                        value={exportConfig.endDate}

                        onChange={(e) =>

                          handleConfigChange('endDate', e.target.value)

                        }

                        max={new Date().toISOString().split('T')[0]}

                        className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500"

                      />

                    </div>

                  </div>

                </div>

              )}

              {/\* Filters \*/}

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

                <h3 className="text-sm font-semibold text-gray-700">

                  Optional Filters

                </h3>

                {/\* Category Filter \*/}

                <div>

                  <label className="block text-sm font-medium text-gray-700 mb-2">

                    Category

                  </label>

                  <select

                    value={exportConfig.categoryId}

                    onChange={(e) =>

                      handleConfigChange('categoryId', e.target.value)

                    }

                    className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500"

                  >

                    <option value="">All Categories</option>

                    {categories.map((cat) => (

                      <option key={cat.id} value={cat.id}>

                        {cat.name}

                      </option>

                    ))}

                  </select>

                </div>

                {/\* Payment Mode Filter \*/}

                <div>

                  <label className="block text-sm font-medium text-gray-700 mb-2">

                    Payment Mode

                  </label>

                  <select

                    value={exportConfig.paymentModeId}

                    onChange={(e) =>

                      handleConfigChange('paymentModeId', e.target.value)

                    }

                    className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500"

                  >

                    <option value="">All Payment Modes</option>

                    {paymentModes.map((pm) => (

                      <option key={pm.id} value={pm.id}>

                        {pm.name}

                        {pm.bank\_name ? ` (${pm.bank\_name})` : ''}

                      </option>

                    ))}

                  </select>

                </div>

              </div>

              {/\* Export Button \*/}

              <div className="mt-6 pt-6 border-t border-gray-200">

                <button

                  onClick={handleExport}

                  disabled={loading}

                  className="w-full flex items-center justify-center gap-3 px-6 py-4 bg-gradient-to-r from-green-600 to-teal-600 text-white rounded-xl font-bold text-lg hover:from-green-700 hover:to-teal-700 disabled:opacity-50 disabled:cursor-not-allowed transition-all shadow-lg"

                >

                  {loading ? (

                    <>

                      <svg

                        className="animate-spin h-6 w-6 text-white"

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

                        fill="none"

                        viewBox="0 0 24 24"

                      >

                        <circle

                          className="opacity-25"

                          cx="12"

                          cy="12"

                          r="10"

                          stroke="currentColor"

                          strokeWidth="4"

                        ></circle>

                        <path

                          className="opacity-75"

                          fill="currentColor"

                          d="M4 12a8 8 0 018-8V0C5.373 0 0 5.373 0 12h4zm2 5.291A7.962 7.962 0 014 12H0c0 3.042 1.135 5.824 3 7.938l3-2.647z"

                        ></path>

                      </svg>

                      Generating {exportConfig.format.toUpperCase()}...

                    </>

                  ) : (

                    <>

                      <MdFileDownload className="text-2xl" />

                      Export {exportConfig.format.toUpperCase()}

                    </>

                  )}

                </button>

              </div>

            </div>

          </div>

          {/\* Sidebar \*/}

          <div className="space-y-6">

            {/\* Export Summary \*/}

            <div className="bg-white rounded-xl shadow-lg p-6">

              <h3 className="text-lg font-bold text-gray-900 mb-4">

                Export Summary

              </h3>

              <div className="space-y-3">

                <div className="flex justify-between items-center py-2 border-b border-gray-200">

                  <span className="text-sm text-gray-600">Format:</span>

                  <span className="text-sm font-bold text-gray-900">

                    {exportConfig.format.toUpperCase()}

                  </span>

                </div>

                {exportConfig.format === 'pdf' && (

                  <div className="flex justify-between items-center py-2 border-b border-gray-200">

                    <span className="text-sm text-gray-600">Report Type:</span>

                    <span className="text-sm font-bold text-gray-900 capitalize">

                      {exportConfig.reportType}

                    </span>

                  </div>

                )}

                <div className="flex justify-between items-center py-2 border-b border-gray-200">

                  <span className="text-sm text-gray-600">Period:</span>

                  <span className="text-sm font-bold text-gray-900">

                    {getPeriodLabel(exportConfig.period)}

                  </span>

                </div>

                {exportConfig.period === 'custom' &&

                  exportConfig.startDate &&

                  exportConfig.endDate && (

                    <div className="py-2 border-b border-gray-200">

                      <span className="text-sm text-gray-600 block mb-1">

                        Date Range:

                      </span>

                      <span className="text-xs font-medium text-gray-900">

                        {formatDate(exportConfig.startDate)} to{' '}

                        {formatDate(exportConfig.endDate)}

                      </span>

                    </div>

                  )}

                {exportConfig.categoryId && (

                  <div className="flex justify-between items-center py-2 border-b border-gray-200">

                    <span className="text-sm text-gray-600">Category:</span>

                    <span className="text-sm font-bold text-gray-900">

                      {categories.find(

                        (c) => c.id === parseInt(exportConfig.categoryId)

                      )?.name || 'N/A'}

                    </span>

                  </div>

                )}

                {exportConfig.paymentModeId && (

                  <div className="flex justify-between items-center py-2">

                    <span className="text-sm text-gray-600">Payment:</span>

                    <span className="text-sm font-bold text-gray-900">

                      {paymentModes.find(

                        (p) => p.id === parseInt(exportConfig.paymentModeId)

                      )?.name || 'N/A'}

                    </span>

                  </div>

                )}

              </div>

            </div>

            {/\* Export History \*/}

            <div className="bg-white rounded-xl shadow-lg p-6">

              <h3 className="text-lg font-bold text-gray-900 mb-4 flex items-center gap-2">

                <MdCalendarToday className="text-blue-600" />

                Recent Exports

              </h3>

              {exportHistory.length > 0 ? (

                <div className="space-y-3 max-h-80 overflow-y-auto">

                  {exportHistory.map((record) => (

                    <div

                      key={record.id}

                      className="p-3 bg-gray-50 rounded-lg border border-gray-200"

                    >

                      <div className="flex items-start justify-between">

                        <div className="flex items-center gap-2">

                          {record.format === 'PDF' ? (

                            <MdPictureAsPdf className="text-xl text-red-600" />

                          ) : (

                            <MdTableChart className="text-xl text-green-600" />

                          )}

                          <div>

                            <p className="text-sm font-medium text-gray-900">

                              {record.format}

                            </p>

                            <p className="text-xs text-gray-500">

                              {getPeriodLabel(record.period)}

                            </p>

                          </div>

                        </div>

                      </div>

                      <p className="text-xs text-gray-400 mt-2">

                        {new Date(record.timestamp).toLocaleString()}

                      </p>

                    </div>

                  ))}

                </div>

              ) : (

                <p className="text-sm text-gray-500 text-center py-4">

                  No export history yet

                </p>

              )}

            </div>

            {/\* Info Card \*/}

            <div className="bg-gradient-to-br from-blue-50 to-purple-50 rounded-xl shadow-lg p-6 border border-blue-200">

              <h3 className="text-lg font-bold text-gray-900 mb-3">

                Export Formats

              </h3>

              <div className="space-y-3 text-sm">

                <div>

                  <p className="font-medium text-blue-900">CSV Format</p>

                  <p className="text-gray-700 text-xs mt-1">

                    Best for data analysis in Excel, Google Sheets, or other

                    spreadsheet tools

                  </p>

                </div>

                <div>

                  <p className="font-medium text-red-900">PDF Format</p>

                  <p className="text-gray-700 text-xs mt-1">

                    Professional reports with charts and summaries, ideal for

                    sharing or printing

                  </p>

                </div>

              </div>

            </div>

          </div>

        </div>

      </div>

    </div>

  );

};

export default ExportReports;

E:\expense\_tracker\frontend\src\pages\layouts

E:\expense\_tracker\frontend\src\pages\layouts\components

E:\expense\_tracker\frontend\src\pages\layouts\AuthLayout.jsx

/\*\*

 \* AuthLayout Component

 \* Layout for authentication pages

 \*/

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

import { MdAccountBalanceWallet } from 'react-icons/md';

const AuthLayout = () => {

  return (

    <div className="min-h-screen bg-gray-50 relative">

      <header className="absolute top-0 left-0 right-0 z-10">

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

          <div className="flex items-center gap-3">

            <div className="w-10 h-10 bg-gradient-to-br from-blue-500 to-purple-600 rounded-xl flex items-center justify-center shadow-lg">

              <MdAccountBalanceWallet className="text-2xl text-white" />

            </div>

            <div>

              <h1 className="text-xl font-bold text-gray-900">Expense Tracker</h1>

              <p className="text-xs text-gray-500">Manage your finances smartly</p>

            </div>

          </div>

        </div>

      </header>

      <main className="min-h-screen flex items-center justify-center pt-20 pb-20">

        <Outlet />

      </main>

      <footer className="absolute bottom-0 left-0 right-0 py-6">

        <div className="text-center text-sm text-gray-600">

          <p>© {new Date().getFullYear()} Expense Tracker. All rights reserved.</p>

        </div>

      </footer>

    </div>

  );

};

export default AuthLayout;

E:\expense\_tracker\frontend\src\pages\layouts\DashboardLayout.jsx

/\*\*

 \* DashboardLayout Component

 \* Specialized layout for dashboard-specific features

 \* Can be used if dashboard needs different layout than other pages

 \* Currently uses same structure as MainLayout

 \*/

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

import MainLayout from './MainLayout';

const DashboardLayout = () => {

  // You can add dashboard-specific features here

  // For now, it uses the MainLayout structure

  // This provides flexibility for future customization

  return <MainLayout />;

};

export default DashboardLayout;

E:\expense\_tracker\frontend\src\pages\layouts\MainLayout.jsx

/\*\*

 \* MainLayout Component

 \* Main authenticated layout with sidebar navigation and header

 \*/

import { useState } from 'react';

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

import { toast } from 'react-hot-toast';

import {

  MdDashboard,

  MdAccountBalanceWallet,

  MdBarChart,

  MdPsychology,

  MdFileDownload,

  MdPerson,

  MdMenu,

  MdClose,

  MdLogout,

  MdNotifications,

  MdSettings,

} from 'react-icons/md';

import { getUserData, logout } from '../../utils/auth'; // ✅ FIXED: Two levels up

const MainLayout = () => {

  const navigate = useNavigate();

  const userData = getUserData();

  const [sidebarOpen, setSidebarOpen] = useState(false);

  const handleLogout = async () => {

    try {

      await logout();

      toast.success('Logged out successfully');

      navigate('/login');

    } catch (error) {

      console.error('Logout error:', error);

      toast.error('Logout failed');

    }

  };

  const navigation = [

    {

      name: 'Dashboard',

      path: '/dashboard',

      icon: MdDashboard,

      activeColor: 'bg-blue-50 text-blue-600 border-blue-600',

    },

    {

      name: 'Expenses',

      path: '/expenses',

      icon: MdAccountBalanceWallet,

      activeColor: 'bg-purple-50 text-purple-600 border-purple-600',

    },

    {

      name: 'Analytics',

      path: '/analytics',

      icon: MdBarChart,

      activeColor: 'bg-green-50 text-green-600 border-green-600',

    },

    {

      name: 'AI Insights',

      path: '/ai-insights',

      icon: MdPsychology,

      activeColor: 'bg-pink-50 text-pink-600 border-pink-600',

    },

    {

      name: 'Export',

      path: '/export',

      icon: MdFileDownload,

      activeColor: 'bg-orange-50 text-orange-600 border-orange-600',

    },

    {

      name: 'Profile',

      path: '/profile',

      icon: MdPerson,

      activeColor: 'bg-gray-50 text-gray-600 border-gray-600',

    },

  ];

  return (

    <div className="min-h-screen bg-gray-50">

      {sidebarOpen && (

        <div

          className="fixed inset-0 bg-black bg-opacity-50 z-40 lg:hidden"

          onClick={() => setSidebarOpen(false)}

        ></div>

      )}

      <aside

        className={`fixed top-0 left-0 z-50 h-full w-72 bg-white shadow-xl transform transition-transform duration-300 ease-in-out lg:translate-x-0 ${

          sidebarOpen ? 'translate-x-0' : '-translate-x-full'

        }`}

      >

        <div className="h-20 flex items-center justify-between px-6 border-b border-gray-200">

          <Link to="/dashboard" className="flex items-center gap-3">

            <div className="w-10 h-10 bg-gradient-to-br from-blue-500 to-purple-600 rounded-xl flex items-center justify-center">

              <MdAccountBalanceWallet className="text-2xl text-white" />

            </div>

            <div>

              <h1 className="text-xl font-bold text-gray-900">Expense</h1>

              <p className="text-xs text-gray-500">Tracker</p>

            </div>

          </Link>

          <button

            onClick={() => setSidebarOpen(false)}

            className="lg:hidden p-2 rounded-lg hover:bg-gray-100 transition-colors"

          >

            <MdClose className="text-2xl text-gray-600" />

          </button>

        </div>

        <div className="p-6 border-b border-gray-200">

          <div className="flex items-center gap-3">

            <div className="w-12 h-12 bg-gradient-to-br from-blue-500 to-purple-600 rounded-full flex items-center justify-center">

              <span className="text-white font-bold text-lg">

                {userData?.fullname?.charAt(0) || userData?.email?.charAt(0) || 'U'}

              </span>

            </div>

            <div className="flex-1 min-w-0">

              <p className="text-sm font-semibold text-gray-900 truncate">

                {userData?.fullname || 'User'}

              </p>

              <p className="text-xs text-gray-500 truncate">{userData?.email}</p>

            </div>

          </div>

        </div>

        <nav className="flex-1 p-4 space-y-2 overflow-y-auto">

          {navigation.map((item) => {

            const Icon = item.icon;

            return (

              <NavLink

                key={item.path}

                to={item.path}

                onClick={() => setSidebarOpen(false)}

                className={({ isActive }) =>

                  `flex items-center gap-3 px-4 py-3 rounded-xl transition-all border-l-4 ${

                    isActive

                      ? item.activeColor + ' font-semibold'

                      : 'border-transparent text-gray-600 hover:bg-gray-50'

                  }`

                }

              >

                <Icon className="text-2xl" />

                <span>{item.name}</span>

              </NavLink>

            );

          })}

        </nav>

        <div className="p-4 border-t border-gray-200">

          <Link

            to="/settings"

            onClick={() => setSidebarOpen(false)}

            className="flex items-center gap-3 px-4 py-3 rounded-xl text-gray-600 hover:bg-gray-50 transition-all mb-2"

          >

            <MdSettings className="text-2xl" />

            <span>Settings</span>

          </Link>

          <button

            onClick={handleLogout}

            className="w-full flex items-center gap-3 px-4 py-3 rounded-xl text-red-600 hover:bg-red-50 transition-all"

          >

            <MdLogout className="text-2xl" />

            <span>Logout</span>

          </button>

        </div>

      </aside>

      <div className="lg:ml-72">

        <header className="h-20 bg-white border-b border-gray-200 sticky top-0 z-30">

          <div className="h-full px-4 sm:px-6 flex items-center justify-between">

            <button

              onClick={() => setSidebarOpen(true)}

              className="lg:hidden p-2 rounded-lg hover:bg-gray-100 transition-colors"

            >

              <MdMenu className="text-2xl text-gray-600" />

            </button>

            <Link to="/dashboard" className="lg:hidden flex items-center gap-2">

              <div className="w-8 h-8 bg-gradient-to-br from-blue-500 to-purple-600 rounded-lg flex items-center justify-center">

                <MdAccountBalanceWallet className="text-lg text-white" />

              </div>

              <span className="font-bold text-gray-900">Expense Tracker</span>

            </Link>

            <div className="hidden lg:block">

              <h2 className="text-xl font-bold text-gray-900">

                Welcome back, {userData?.fullname || 'User'}!

              </h2>

              <p className="text-sm text-gray-500">Manage your expenses efficiently</p>

            </div>

            <div className="flex items-center gap-3">

              <button className="relative p-2 rounded-lg hover:bg-gray-100 transition-colors">

                <MdNotifications className="text-2xl text-gray-600" />

                <span className="absolute top-1 right-1 w-2 h-2 bg-red-500 rounded-full"></span>

              </button>

              <Link

                to="/profile"

                className="hidden sm:flex items-center gap-2 px-4 py-2 rounded-lg hover:bg-gray-100 transition-colors"

              >

                <div className="w-8 h-8 bg-gradient-to-br from-blue-500 to-purple-600 rounded-full flex items-center justify-center">

                  <span className="text-white font-semibold text-sm">

                    {userData?.fullname?.charAt(0) || userData?.email?.charAt(0) || 'U'}

                  </span>

                </div>

                <span className="text-sm font-medium text-gray-700">Profile</span>

              </Link>

            </div>

          </div>

        </header>

        <main className="min-h-[calc(100vh-5rem)]">

          <Outlet />

        </main>

        <footer className="bg-white border-t border-gray-200 py-6">

          <div className="px-6 text-center text-sm text-gray-600">

            <p>© {new Date().getFullYear()} Expense Tracker. Built with React & Tailwind CSS.</p>

          </div>

        </footer>

      </div>

    </div>

  );

};

export default MainLayout;

E:\expense\_tracker\frontend\src\pages\layouts\components

E:\expense\_tracker\frontend\src\pages\layouts\components\Breadcrumb.jsx

/\*\*

 \* Breadcrumb Component

 \* Navigation breadcrumb showing current page hierarchy

 \*/

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

import { MdChevronRight, MdHome } from 'react-icons/md';

const Breadcrumb = () => {

  const location = useLocation();

  const pathnames = location.pathname.split('/').filter((x) => x);

  // Mapping of URL segments to display names

  const breadcrumbNameMap = {

    dashboard: 'Dashboard',

    expenses: 'Expenses',

    add: 'Add Expense',

    edit: 'Edit Expense',

    analytics: 'Analytics',

    category: 'Category Breakdown',

    payment: 'Payment Analysis',

    trends: 'Trend Analysis',

    'ai-insights': 'AI Insights',

    export: 'Export Reports',

    profile: 'Profile',

    settings: 'Account Settings',

  };

  // Don't show breadcrumb on home/dashboard or auth pages

  if (pathnames.length === 0 || pathnames[0] === 'login' || pathnames[0] === 'register') {

    return null;

  }

  return (

    <nav className="flex items-center gap-2 px-6 py-4 bg-white border-b border-gray-200">

      {/\* Home Link \*/}

      <Link

        to="/dashboard"

        className="flex items-center gap-1 text-gray-500 hover:text-blue-600 transition-colors group"

        aria-label="Home"

      >

        <MdHome className="text-lg group-hover:scale-110 transition-transform" />

        <span className="text-sm font-medium hidden sm:inline">Home</span>

      </Link>

      {/\* Breadcrumb Items \*/}

      {pathnames.map((name, index) => {

        const routeTo = `/${pathnames.slice(0, index + 1).join('/')}`;

        const isLast = index === pathnames.length - 1;

        const displayName = breadcrumbNameMap[name] || name.charAt(0).toUpperCase() + name.slice(1);

        // Skip numeric IDs in breadcrumb display

        const isNumeric = !isNaN(name);

        if (isNumeric) {

          return null;

        }

        return (

          <div key={name} className="flex items-center gap-2">

            <MdChevronRight className="text-gray-400 text-lg" />

            {isLast ? (

              <span className="text-sm font-semibold text-gray-900">

                {displayName}

              </span>

            ) : (

              <Link

                to={routeTo}

                className="text-sm font-medium text-gray-500 hover:text-blue-600 transition-colors"

              >

                {displayName}

              </Link>

            )}

          </div>

        );

      })}

    </nav>

  );

};

export default Breadcrumb;

E:\expense\_tracker\frontend\src\pages\layouts\components\Footer.jsx

/\*\*

 \* Footer Component

 \* Reusable footer with copyright and links

 \*/

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

import {

  MdFavorite,

  MdPrivacyTip,

  MdGavel,

  MdHelp,

  MdEmail,

} from 'react-icons/md';

const Footer = () => {

  const currentYear = new Date().getFullYear();

  return (

    <footer className="bg-white border-t border-gray-200">

      {/\* Main Footer Content \*/}

      <div className="max-w-7xl mx-auto px-6 py-8">

        <div className="grid grid-cols-1 md:grid-cols-4 gap-8">

          {/\* About Section \*/}

          <div className="col-span-1 md:col-span-2">

            <div className="flex items-center gap-2 mb-4">

              <div className="w-8 h-8 bg-gradient-to-br from-blue-500 to-purple-600 rounded-lg flex items-center justify-center">

                <span className="text-white font-bold text-sm">ET</span>

              </div>

              <h3 className="text-lg font-bold text-gray-900">Expense Tracker</h3>

            </div>

            <p className="text-sm text-gray-600 mb-4">

              Manage your personal finances with ease. Track expenses, analyze

              spending patterns, and get AI-powered insights to make better

              financial decisions.

            </p>

            <div className="flex items-center gap-1 text-sm text-gray-600">

              <span>Made with</span>

              <MdFavorite className="text-red-500" />

              <span>using React & Tailwind CSS</span>

            </div>

          </div>

          {/\* Quick Links \*/}

          <div>

            <h4 className="font-semibold text-gray-900 mb-4">Quick Links</h4>

            <ul className="space-y-2">

              <li>

                <Link

                  to="/dashboard"

                  className="text-sm text-gray-600 hover:text-blue-600 transition-colors flex items-center gap-2"

                >

                  Dashboard

                </Link>

              </li>

              <li>

                <Link

                  to="/expenses"

                  className="text-sm text-gray-600 hover:text-blue-600 transition-colors flex items-center gap-2"

                >

                  Expenses

                </Link>

              </li>

              <li>

                <Link

                  to="/analytics"

                  className="text-sm text-gray-600 hover:text-blue-600 transition-colors flex items-center gap-2"

                >

                  Analytics

                </Link>

              </li>

              <li>

                <Link

                  to="/ai-insights"

                  className="text-sm text-gray-600 hover:text-blue-600 transition-colors flex items-center gap-2"

                >

                  AI Insights

                </Link>

              </li>

            </ul>

          </div>

          {/\* Support & Legal \*/}

          <div>

            <h4 className="font-semibold text-gray-900 mb-4">Support</h4>

            <ul className="space-y-2">

              <li>

                <button className="text-sm text-gray-600 hover:text-blue-600 transition-colors flex items-center gap-2">

                  <MdHelp className="text-base" />

                  Help Center

                </button>

              </li>

              <li>

                <button className="text-sm text-gray-600 hover:text-blue-600 transition-colors flex items-center gap-2">

                  <MdEmail className="text-base" />

                  Contact Us

                </button>

              </li>

              <li>

                <button className="text-sm text-gray-600 hover:text-blue-600 transition-colors flex items-center gap-2">

                  <MdPrivacyTip className="text-base" />

                  Privacy Policy

                </button>

              </li>

              <li>

                <button className="text-sm text-gray-600 hover:text-blue-600 transition-colors flex items-center gap-2">

                  <MdGavel className="text-base" />

                  Terms of Service

                </button>

              </li>

            </ul>

          </div>

        </div>

      </div>

      {/\* Bottom Bar \*/}

      <div className="border-t border-gray-200 bg-gray-50">

        <div className="max-w-7xl mx-auto px-6 py-4">

          <div className="flex flex-col md:flex-row items-center justify-between gap-4">

            <p className="text-sm text-gray-600">

              © {currentYear} Expense Tracker. All rights reserved.

            </p>

            <div className="flex items-center gap-4">

              <span className="text-xs text-gray-500">Version 1.0.0</span>

              <div className="flex items-center gap-1">

                <span className="w-2 h-2 bg-green-500 rounded-full animate-pulse"></span>

                <span className="text-xs text-gray-600">All systems operational</span>

              </div>

            </div>

          </div>

        </div>

      </div>

    </footer>

  );

};

export default Footer;

E:\expense\_tracker\frontend\src\pages\layouts\components\Header.jsx

/\*\*

 \* Header Component

 \* Reusable header with navigation actions, notifications, and user menu

 \*/

import { useState, useEffect } from 'react';

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

import {

  MdMenu,

  MdNotifications,

  MdSettings,

  MdPerson,

  MdLogout,

  MdAccountBalanceWallet,

} from 'react-icons/md';

import { getUserData, logout } from '../../utils/auth';

import { toast } from 'react-hot-toast';

const Header = ({ onMenuClick }) => {

  const navigate = useNavigate();

  const location = useLocation(); // ✅ ADDED: Track route changes

  const [userData, setUserData] = useState(null);

  const [showUserMenu, setShowUserMenu] = useState(false);

  const [showNotifications, setShowNotifications] = useState(false);

  // ✅ FIXED: Reload user data when route changes

  useEffect(() => {

    const user = getUserData();

    console.log('📋 Header - User Data from localStorage:', user);

    console.log('📋 Full Name Value:', user?.full\_name);

    console.log('📋 Email Value:', user?.email);

    setUserData(user);

  }, [location]); // ✅ CHANGED: Added location as dependency

  const handleLogout = async () => {

    try {

      await logout();

      toast.success('Logged out successfully');

      navigate('/login');

    } catch (error) {

      console.error('Logout error:', error);

      toast.error('Logout failed');

    }

  };

  // Sample notifications (replace with real data)

  const notifications = [

    {

      id: 1,

      title: 'New expense added',

      message: 'Food expense of ₹500 was added',

      time: '5 minutes ago',

      read: false,

    },

    {

      id: 2,

      title: 'Monthly report ready',

      message: 'Your expense report for October is ready',

      time: '2 hours ago',

      read: false,

    },

    {

      id: 3,

      title: 'Budget alert',

      message: 'You have reached 80% of your monthly budget',

      time: '1 day ago',

      read: true,

    },

  ];

  const unreadCount = notifications.filter((n) => !n.read).length;

  // Get display name with fallback

  const getDisplayName = () => {

    if (userData?.full\_name && userData.full\_name.trim()) {

      return userData.full\_name;

    }

    if (userData?.email) {

      return userData.email.split('@')[0];

    }

    return 'User';

  };

  // Get first name

  const getFirstName = () => {

    const fullName = getDisplayName();

    return fullName.split(' ')[0];

  };

  // Get initial for avatar

  const getInitial = () => {

    if (userData?.full\_name && userData.full\_name.trim()) {

      return userData.full\_name.charAt(0).toUpperCase();

    }

    if (userData?.email) {

      return userData.email.charAt(0).toUpperCase();

    }

    return 'U';

  };

  return (

    <header className="h-20 bg-white border-b border-gray-200 sticky top-0 z-30 shadow-sm">

      <div className="h-full px-4 sm:px-6 flex items-center justify-between">

        {/\* Left Section \*/}

        <div className="flex items-center gap-4">

          {/\* Mobile Menu Button \*/}

          <button

            onClick={onMenuClick}

            className="lg:hidden p-2 rounded-lg hover:bg-gray-100 transition-colors"

            aria-label="Open menu"

          >

            <MdMenu className="text-2xl text-gray-600" />

          </button>

          {/\* Logo for Mobile \*/}

          <Link

            to="/dashboard"

            className="lg:hidden flex items-center gap-2"

          >

            <div className="w-8 h-8 bg-gradient-to-br from-blue-500 to-purple-600 rounded-lg flex items-center justify-center">

              <MdAccountBalanceWallet className="text-lg text-white" />

            </div>

            <span className="font-bold text-gray-900">Expense Tracker</span>

          </Link>

          {/\* Welcome Message for Desktop \*/}

          <div className="hidden lg:block">

            <h2 className="text-xl font-bold text-gray-900">

              Welcome back, {getFirstName()}!

            </h2>

            <p className="text-sm text-gray-500">

              Track and manage your expenses efficiently

            </p>

          </div>

        </div>

        {/\* Right Section - Actions \*/}

        <div className="flex items-center gap-2">

          {/\* Notifications Dropdown \*/}

          <div className="relative">

            <button

              onClick={() => setShowNotifications(!showNotifications)}

              className="relative p-2 rounded-lg hover:bg-gray-100 transition-colors"

              aria-label="Notifications"

            >

              <MdNotifications className="text-2xl text-gray-600" />

              {unreadCount > 0 && (

                <span className="absolute top-1 right-1 w-5 h-5 bg-red-500 text-white text-xs font-bold rounded-full flex items-center justify-center">

                  {unreadCount}

                </span>

              )}

            </button>

            {/\* Notifications Dropdown Menu \*/}

            {showNotifications && (

              <>

                <div

                  className="fixed inset-0 z-40"

                  onClick={() => setShowNotifications(false)}

                ></div>

                <div className="absolute right-0 mt-2 w-80 bg-white rounded-xl shadow-2xl border border-gray-200 z-50">

                  <div className="p-4 border-b border-gray-200">

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

                      <h3 className="font-bold text-gray-900">Notifications</h3>

                      {unreadCount > 0 && (

                        <span className="px-2 py-1 bg-red-100 text-red-600 text-xs font-medium rounded-full">

                          {unreadCount} new

                        </span>

                      )}

                    </div>

                  </div>

                  <div className="max-h-96 overflow-y-auto">

                    {notifications.length > 0 ? (

                      notifications.map((notification) => (

                        <div

                          key={notification.id}

                          className={`p-4 border-b border-gray-100 hover:bg-gray-50 transition-colors cursor-pointer ${

                            !notification.read ? 'bg-blue-50' : ''

                          }`}

                        >

                          <div className="flex items-start gap-3">

                            <div

                              className={`w-2 h-2 rounded-full mt-2 ${

                                !notification.read ? 'bg-blue-600' : 'bg-gray-300'

                              }`}

                            ></div>

                            <div className="flex-1">

                              <p className="text-sm font-semibold text-gray-900">

                                {notification.title}

                              </p>

                              <p className="text-xs text-gray-600 mt-1">

                                {notification.message}

                              </p>

                              <p className="text-xs text-gray-400 mt-1">

                                {notification.time}

                              </p>

                            </div>

                          </div>

                        </div>

                      ))

                    ) : (

                      <div className="p-8 text-center text-gray-500">

                        <p>No notifications</p>

                      </div>

                    )}

                  </div>

                  <div className="p-3 border-t border-gray-200 text-center">

                    <button className="text-sm text-blue-600 hover:text-blue-700 font-medium">

                      View all notifications

                    </button>

                  </div>

                </div>

              </>

            )}

          </div>

          {/\* Settings Link \*/}

          <Link

            to="/settings"

            className="hidden sm:block p-2 rounded-lg hover:bg-gray-100 transition-colors"

            aria-label="Settings"

          >

            <MdSettings className="text-2xl text-gray-600" />

          </Link>

          {/\* User Profile Dropdown \*/}

          <div className="relative">

            <button

              onClick={() => setShowUserMenu(!showUserMenu)}

              className="flex items-center gap-2 px-3 py-2 rounded-lg hover:bg-gray-100 transition-colors"

            >

              <div className="w-9 h-9 bg-gradient-to-br from-blue-500 to-purple-600 rounded-full flex items-center justify-center">

                <span className="text-white font-semibold text-sm">

                  {getInitial()}

                </span>

              </div>

              <span className="hidden sm:block text-sm font-medium text-gray-700">

                {getFirstName()}

              </span>

            </button>

            {/\* User Dropdown Menu \*/}

            {showUserMenu && (

              <>

                <div

                  className="fixed inset-0 z-40"

                  onClick={() => setShowUserMenu(false)}

                ></div>

                <div className="absolute right-0 mt-2 w-64 bg-white rounded-xl shadow-2xl border border-gray-200 z-50">

                  {/\* User Info \*/}

                  <div className="p-4 border-b border-gray-200">

                    <div className="flex items-center gap-3">

                      <div className="w-12 h-12 bg-gradient-to-br from-blue-500 to-purple-600 rounded-full flex items-center justify-center">

                        <span className="text-white font-bold text-lg">

                          {getInitial()}

                        </span>

                      </div>

                      <div className="flex-1 min-w-0">

                        <p className="text-sm font-semibold text-gray-900 truncate">

                          {getDisplayName()}

                        </p>

                        <p className="text-xs text-gray-500 truncate">

                          {userData?.email}

                        </p>

                      </div>

                    </div>

                  </div>

                  {/\* Menu Items \*/}

                  <div className="p-2">

                    <Link

                      to="/profile"

                      onClick={() => setShowUserMenu(false)}

                      className="flex items-center gap-3 px-4 py-3 rounded-lg text-gray-700 hover:bg-gray-50 transition-colors"

                    >

                      <MdPerson className="text-xl" />

                      <span className="text-sm font-medium">My Profile</span>

                    </Link>

                    <Link

                      to="/settings"

                      onClick={() => setShowUserMenu(false)}

                      className="flex items-center gap-3 px-4 py-3 rounded-lg text-gray-700 hover:bg-gray-50 transition-colors"

                    >

                      <MdSettings className="text-xl" />

                      <span className="text-sm font-medium">Settings</span>

                    </Link>

                  </div>

                  {/\* Logout \*/}

                  <div className="p-2 border-t border-gray-200">

                    <button

                      onClick={handleLogout}

                      className="w-full flex items-center gap-3 px-4 py-3 rounded-lg text-red-600 hover:bg-red-50 transition-colors"

                    >

                      <MdLogout className="text-xl" />

                      <span className="text-sm font-medium">Logout</span>

                    </button>

                  </div>

                </div>

              </>

            )}

          </div>

        </div>

      </div>

    </header>

  );

};

export default Header;

E:\expense\_tracker\frontend\src\pages\layouts\components\index.js

/\*\*

 \* Layout Components Index

 \* Central export point for all layout components

 \*/

export { default as Header } from './Header';

export { default as Sidebar } from './Sidebar';

export { default as Footer } from './Footer';

export { default as Breadcrumb } from './Breadcrumb';

E:\expense\_tracker\frontend\src\pages\layouts\components\Sidebar.jsx

/\*\*

 \* Sidebar Component

 \* Reusable navigation sidebar with responsive design

 \*/

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

import {

  MdDashboard,

  MdAccountBalanceWallet,

  MdBarChart,

  MdPsychology,

  MdFileDownload,

  MdPerson,

  MdClose,

  MdSettings,

  MdLogout,

  MdKeyboardArrowDown,

} from 'react-icons/md';

import { useState, useEffect } from 'react';

import { getUserData, logout } from '../../utils/auth';

import { toast } from 'react-hot-toast';

const Sidebar = ({ isOpen, onClose }) => {

  const navigate = useNavigate();

  const location = useLocation(); // ✅ ADDED: Track route changes

  const [userData, setUserData] = useState(null); // ✅ CHANGED: Now using state

  const [analyticsExpanded, setAnalyticsExpanded] = useState(false);

  // ✅ ADDED: Reload user data when route changes

  useEffect(() => {

    const user = getUserData();

    console.log('📋 Sidebar - User Data from localStorage:', user);

    console.log('📋 Sidebar - Full Name:', user?.full\_name);

    console.log('📋 Sidebar - Email:', user?.email);

    setUserData(user);

  }, [location]);

  const handleLogout = async () => {

    try {

      await logout();

      toast.success('Logged out successfully');

      navigate('/login');

    } catch (error) {

      console.error('Logout error:', error);

      toast.error('Logout failed');

    }

  };

  // ✅ ADDED: Helper function to get display name with fallback

  const getDisplayName = () => {

    if (userData?.full\_name && userData.full\_name.trim()) {

      return userData.full\_name;

    }

    if (userData?.email) {

      return userData.email.split('@')[0];

    }

    return 'User';

  };

  // ✅ ADDED: Helper function to get initial for avatar

  const getInitial = () => {

    if (userData?.full\_name && userData.full\_name.trim()) {

      return userData.full\_name.charAt(0).toUpperCase();

    }

    if (userData?.email) {

      return userData.email.charAt(0).toUpperCase();

    }

    return 'U';

  };

  const navigation = [

    {

      name: 'Dashboard',

      path: '/dashboard',

      icon: MdDashboard,

      color: 'text-blue-600',

      hoverBg: 'hover:bg-blue-50',

      activeBg: 'bg-blue-50',

      activeColor: 'text-blue-600',

      activeBorder: 'border-blue-600',

    },

    {

      name: 'Expenses',

      path: '/expenses',

      icon: MdAccountBalanceWallet,

      color: 'text-purple-600',

      hoverBg: 'hover:bg-purple-50',

      activeBg: 'bg-purple-50',

      activeColor: 'text-purple-600',

      activeBorder: 'border-purple-600',

    },

    {

      name: 'Analytics',

      path: '/analytics',

      icon: MdBarChart,

      color: 'text-green-600',

      hoverBg: 'hover:bg-green-50',

      activeBg: 'bg-green-50',

      activeColor: 'text-green-600',

      activeBorder: 'border-green-600',

      submenu: [

        { name: 'Overview', path: '/analytics' },

        { name: 'Category Breakdown', path: '/analytics/category' },

        { name: 'Payment Analysis', path: '/analytics/payment' },

        { name: 'Trends', path: '/analytics/trends' },

      ],

    },

    {

      name: 'AI Insights',

      path: '/ai-insights',

      icon: MdPsychology,

      color: 'text-pink-600',

      hoverBg: 'hover:bg-pink-50',

      activeBg: 'bg-pink-50',

      activeColor: 'text-pink-600',

      activeBorder: 'border-pink-600',

    },

    {

      name: 'Export',

      path: '/export',

      icon: MdFileDownload,

      color: 'text-orange-600',

      hoverBg: 'hover:bg-orange-50',

      activeBg: 'bg-orange-50',

      activeColor: 'text-orange-600',

      activeBorder: 'border-orange-600',

    },

    {

      name: 'Profile',

      path: '/profile',

      icon: MdPerson,

      color: 'text-gray-600',

      hoverBg: 'hover:bg-gray-50',

      activeBg: 'bg-gray-50',

      activeColor: 'text-gray-600',

      activeBorder: 'border-gray-600',

    },

  ];

  return (

    <>

      {/\* Overlay for mobile \*/}

      {isOpen && (

        <div

          className="fixed inset-0 bg-black bg-opacity-50 z-40 lg:hidden"

          onClick={onClose}

        ></div>

      )}

      {/\* Sidebar \*/}

      <aside

        className={`fixed top-0 left-0 z-50 h-full w-72 bg-white shadow-2xl transform transition-transform duration-300 ease-in-out lg:translate-x-0 flex flex-col ${

          isOpen ? 'translate-x-0' : '-translate-x-full'

        }`}

      >

        {/\* Sidebar Header \*/}

        <div className="h-20 flex items-center justify-between px-6 border-b border-gray-200 flex-shrink-0">

          <Link to="/dashboard" className="flex items-center gap-3" onClick={onClose}>

            <div className="w-10 h-10 bg-gradient-to-br from-blue-500 to-purple-600 rounded-xl flex items-center justify-center shadow-lg">

              <MdAccountBalanceWallet className="text-2xl text-white" />

            </div>

            <div>

              <h1 className="text-xl font-bold text-gray-900">Expense</h1>

              <p className="text-xs text-gray-500">Tracker</p>

            </div>

          </Link>

          <button

            onClick={onClose}

            className="lg:hidden p-2 rounded-lg hover:bg-gray-100 transition-colors"

            aria-label="Close menu"

          >

            <MdClose className="text-2xl text-gray-600" />

          </button>

        </div>

        {/\* User Info \*/}

        <div className="p-6 border-b border-gray-200 flex-shrink-0">

          <Link

            to="/profile"

            onClick={onClose}

            className="flex items-center gap-3 hover:bg-gray-50 -mx-2 px-2 py-2 rounded-lg transition-colors"

          >

            <div className="w-12 h-12 bg-gradient-to-br from-blue-500 to-purple-600 rounded-full flex items-center justify-center flex-shrink-0">

              <span className="text-white font-bold text-lg">

                {getInitial()}

              </span>

            </div>

            <div className="flex-1 min-w-0">

              <p className="text-sm font-semibold text-gray-900 truncate">

                {getDisplayName()}

              </p>

              <p className="text-xs text-gray-500 truncate">{userData?.email}</p>

            </div>

          </Link>

        </div>

        {/\* Navigation \*/}

        <nav className="flex-1 p-4 space-y-2 overflow-y-auto">

          {navigation.map((item) => {

            const Icon = item.icon;

            const hasSubmenu = item.submenu && item.submenu.length > 0;

            return (

              <div key={item.path}>

                {hasSubmenu ? (

                  <>

                    <button

                      onClick={() => setAnalyticsExpanded(!analyticsExpanded)}

                      className="w-full flex items-center justify-between gap-3 px-4 py-3 rounded-xl transition-all text-gray-600 hover:bg-gray-50"

                    >

                      <div className="flex items-center gap-3">

                        <Icon className="text-2xl" />

                        <span>{item.name}</span>

                      </div>

                      <MdKeyboardArrowDown

                        className={`text-xl transition-transform ${

                          analyticsExpanded ? 'rotate-180' : ''

                        }`}

                      />

                    </button>

                    {analyticsExpanded && (

                      <div className="ml-8 mt-2 space-y-1">

                        {item.submenu.map((subitem) => (

                          <NavLink

                            key={subitem.path}

                            to={subitem.path}

                            onClick={onClose}

                            className={({ isActive }) =>

                              `block px-4 py-2 rounded-lg text-sm transition-all ${

                                isActive

                                  ? 'bg-green-50 text-green-600 font-medium'

                                  : 'text-gray-600 hover:bg-gray-50'

                              }`

                            }

                          >

                            {subitem.name}

                          </NavLink>

                        ))}

                      </div>

                    )}

                  </>

                ) : (

                  <NavLink

                    to={item.path}

                    onClick={onClose}

                    className={({ isActive }) =>

                      `flex items-center gap-3 px-4 py-3 rounded-xl transition-all border-l-4 ${

                        isActive

                          ? `${item.activeBg} ${item.activeColor} ${item.activeBorder} font-semibold`

                          : `border-transparent text-gray-600 ${item.hoverBg}`

                      }`

                    }

                  >

                    <Icon className="text-2xl" />

                    <span>{item.name}</span>

                  </NavLink>

                )}

              </div>

            );

          })}

        </nav>

        {/\* Sidebar Footer \*/}

        <div className="p-4 border-t border-gray-200 flex-shrink-0 space-y-2">

          <Link

            to="/settings"

            onClick={onClose}

            className="flex items-center gap-3 px-4 py-3 rounded-xl text-gray-600 hover:bg-gray-50 transition-all"

          >

            <MdSettings className="text-2xl" />

            <span>Settings</span>

          </Link>

          <button

            onClick={() => {

              onClose();

              handleLogout();

            }}

            className="w-full flex items-center gap-3 px-4 py-3 rounded-xl text-red-600 hover:bg-red-50 transition-all"

          >

            <MdLogout className="text-2xl" />

            <span>Logout</span>

          </button>

        </div>

      </aside>

    </>

  );

};

export default Sidebar;

E:\expense\_tracker\frontend\src\pages\profile\Profile.jsx

/\*\*

 \* Profile Component

 \* User profile page with account information and statistics

 \* Backend: User model with email, fullname, created\_at, updated\_at

 \*/

import { useState, useEffect } from 'react';

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

import { toast } from 'react-hot-toast';

import {

  MdPerson,

  MdEmail,

  MdCalendarToday,

  MdEdit,

  MdTrendingUp,

  MdAccountBalanceWallet,

  MdBarChart,

  MdSettings,

} from 'react-icons/md';

import { getUserData } from '../../utils/auth';

import { expenseAPI, analyticsAPI } from '../../services/api';

import { formatCurrency, formatDate } from '../../utils/helpers';

const Profile = () => {

  const location = useLocation(); // ✅ ADDED: Track route changes

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

  const [userData, setUserData] = useState(null);

  const [statistics, setStatistics] = useState({

    totalExpenses: 0,

    expenseCount: 0,

    averageExpense: 0,

    topCategory: null,

    mostUsedPayment: null,

  });

  useEffect(() => {

    fetchProfileData();

  }, [location]); // ✅ CHANGED: Added location as dependency

  const fetchProfileData = async () => {

    try {

      setLoading(true);

      // Get user data from localStorage

      const user = getUserData();

      console.log('🔍 Profile - User Data from localStorage:', user);

      console.log('🔍 Profile - Full Name:', user?.full\_name);

      console.log('🔍 Profile - Email:', user?.email);

      setUserData(user);

      // Fetch user statistics

      const today = new Date();

      const thirtyDaysAgo = new Date();

      thirtyDaysAgo.setDate(today.getDate() - 30);

      const startDate = thirtyDaysAgo.toISOString().split('T')[0];

      const endDate = today.toISOString().split('T')[0];

      // ✅ FIXED: Use Promise.allSettled for partial failure handling

      const results = await Promise.allSettled([

        analyticsAPI.getCategoryBreakdown(startDate, endDate),

        analyticsAPI.getPaymentBreakdown(startDate, endDate),

        expenseAPI.getExpenses({ start\_date: startDate, end\_date: endDate }),

      ]);

      // ✅ FIXED: Safe data access with fallbacks

      const categories =

        results[0].status === 'fulfilled'

          ? results[0].value.data?.data || []

          : [];

      const payments =

        results[1].status === 'fulfilled'

          ? results[1].value.data?.data || []

          : [];

      // ✅ FIXED: Safe calculations

      const totalExpenses = categories.reduce(

        (sum, cat) => sum + (cat.total || 0),

        0

      );

      const expenseCount = categories.reduce(

        (sum, cat) => sum + (cat.count || 0),

        0

      );

      setStatistics({

        totalExpenses,

        expenseCount,

        averageExpense: expenseCount > 0 ? totalExpenses / expenseCount : 0,

        topCategory: categories.length > 0 ? categories[0] : null,

        mostUsedPayment: payments.length > 0 ? payments[0] : null,

      });

    } catch (error) {

      console.error('Error fetching profile data:', error);

      toast.error('Failed to load profile data');

    } finally {

      setLoading(false);

    }

  };

  // ✅ ADDED: Helper function to get display name with fallback

  const getDisplayName = () => {

    if (userData?.full\_name && userData.full\_name.trim()) {

      return userData.full\_name;

    }

    if (userData?.email) {

      return userData.email.split('@')[0];

    }

    return 'User';

  };

  // ✅ ADDED: Helper function to get initial for avatar

  const getInitial = () => {

    if (userData?.full\_name && userData.full\_name.trim()) {

      return userData.full\_name.charAt(0).toUpperCase();

    }

    if (userData?.email) {

      return userData.email.charAt(0).toUpperCase();

    }

    return 'U';

  };

  if (loading) {

    return (

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

        <div className="text-center">

          <div className="animate-spin rounded-full h-16 w-16 border-b-2 border-blue-600 mx-auto"></div>

          <p className="mt-4 text-gray-600">Loading profile...</p>

        </div>

      </div>

    );

  }

  return (

    <div className="min-h-screen bg-gray-50 p-6">

      <div className="max-w-7xl mx-auto">

        {/\* Header \*/}

        <div className="mb-8">

          <h1 className="text-3xl font-bold text-gray-900">Profile</h1>

          <p className="mt-1 text-sm text-gray-600">

            Manage your account information and view your statistics

          </p>

        </div>

        <div className="grid grid-cols-1 lg:grid-cols-3 gap-6">

          {/\* Profile Card \*/}

          <div className="lg:col-span-1">

            <div className="bg-white rounded-xl shadow-lg overflow-hidden">

              {/\* Profile Header \*/}

              <div className="bg-gradient-to-br from-blue-500 to-purple-600 p-8 text-center">

                <div className="w-24 h-24 bg-white rounded-full flex items-center justify-center mx-auto mb-4">

                  <span className="text-5xl text-blue-600 font-bold">

                    {getInitial()}

                  </span>

                </div>

                <h2 className="text-2xl font-bold text-white">

                  {getDisplayName()}

                </h2>

                <p className="text-blue-100 text-sm mt-1">Expense Tracker Member</p>

              </div>

              {/\* Profile Details \*/}

              <div className="p-6 space-y-4">

                <div className="flex items-center gap-3 p-3 bg-gray-50 rounded-lg">

                  <div className="w-10 h-10 bg-blue-100 rounded-full flex items-center justify-center">

                    <MdEmail className="text-xl text-blue-600" />

                  </div>

                  <div className="flex-1">

                    <p className="text-xs text-gray-500">Email Address</p>

                    <p className="text-sm font-medium text-gray-900 truncate">

                      {userData?.email}

                    </p>

                  </div>

                </div>

                <div className="flex items-center gap-3 p-3 bg-gray-50 rounded-lg">

                  <div className="w-10 h-10 bg-green-100 rounded-full flex items-center justify-center">

                    <MdCalendarToday className="text-xl text-green-600" />

                  </div>

                  <div className="flex-1">

                    <p className="text-xs text-gray-500">Member Since</p>

                    <p className="text-sm font-medium text-gray-900">

                      {formatDate(userData?.created\_at)}

                    </p>

                  </div>

                </div>

                <div className="flex items-center gap-3 p-3 bg-gray-50 rounded-lg">

                  <div className="w-10 h-10 bg-purple-100 rounded-full flex items-center justify-center">

                    <MdCalendarToday className="text-xl text-purple-600" />

                  </div>

                  <div className="flex-1">

                    <p className="text-xs text-gray-500">Last Updated</p>

                    <p className="text-sm font-medium text-gray-900">

                      {formatDate(userData?.updated\_at)}

                    </p>

                  </div>

                </div>

                <div className="pt-4 border-t border-gray-200">

                  <div className="flex items-center justify-between mb-2">

                    <span className="text-sm text-gray-600">Account Status</span>

                    <span className="px-3 py-1 bg-green-100 text-green-700 rounded-full text-xs font-medium">

                      {userData?.is\_active ? 'Active' : 'Inactive'}

                    </span>

                  </div>

                </div>

              </div>

              {/\* Action Buttons \*/}

              <div className="p-6 bg-gray-50 border-t border-gray-200">

                <Link

                  to="/settings"

                  className="w-full flex items-center justify-center gap-2 px-4 py-3 bg-blue-600 text-white rounded-lg font-medium hover:bg-blue-700 transition-colors"

                >

                  <MdSettings className="text-xl" />

                  Account Settings

                </Link>

              </div>

            </div>

          </div>

          {/\* Statistics Section \*/}

          <div className="lg:col-span-2 space-y-6">

            {/\* Summary Cards \*/}

            <div className="grid grid-cols-1 md:grid-cols-3 gap-6">

              <div className="bg-gradient-to-br from-blue-500 to-blue-600 rounded-xl shadow-lg p-6 text-white">

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

                  <div>

                    <p className="text-blue-100 text-sm font-medium">

                      Total Spent (30d)

                    </p>

                    <h3 className="text-3xl font-bold mt-2">

                      {formatCurrency(statistics.totalExpenses)}

                    </h3>

                  </div>

                  <div className="bg-blue-400 bg-opacity-50 rounded-full p-3">

                    <MdAccountBalanceWallet className="text-3xl" />

                  </div>

                </div>

              </div>

              <div className="bg-gradient-to-br from-purple-500 to-purple-600 rounded-xl shadow-lg p-6 text-white">

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

                  <div>

                    <p className="text-purple-100 text-sm font-medium">

                      Transactions

                    </p>

                    <h3 className="text-3xl font-bold mt-2">

                      {statistics.expenseCount}

                    </h3>

                  </div>

                  <div className="bg-purple-400 bg-opacity-50 rounded-full p-3">

                    <MdBarChart className="text-3xl" />

                  </div>

                </div>

              </div>

              <div className="bg-gradient-to-br from-green-500 to-green-600 rounded-xl shadow-lg p-6 text-white">

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

                  <div>

                    <p className="text-green-100 text-sm font-medium">Average</p>

                    <h3 className="text-3xl font-bold mt-2">

                      {formatCurrency(statistics.averageExpense)}

                    </h3>

                  </div>

                  <div className="bg-green-400 bg-opacity-50 rounded-full p-3">

                    <MdTrendingUp className="text-3xl" />

                  </div>

                </div>

              </div>

            </div>

            {/\* Top Category & Payment \*/}

            <div className="grid grid-cols-1 md:grid-cols-2 gap-6">

              {/\* Top Category \*/}

              <div className="bg-white rounded-xl shadow-lg p-6">

                <h3 className="text-lg font-bold text-gray-900 mb-4">

                  Top Category

                </h3>

                {statistics.topCategory ? (

                  <div className="space-y-3">

                    <div className="flex items-center gap-3">

                      <div

                        className="w-12 h-12 rounded-lg flex items-center justify-center"

                        style={{

                          backgroundColor:

                            (statistics.topCategory.color || '#3B82F6') + '20',

                        }}

                      >

                        <MdBarChart

                          className="text-2xl"

                          style={{

                            color: statistics.topCategory.color || '#3B82F6',

                          }}

                        />

                      </div>

                      <div className="flex-1">

                        <p className="text-xl font-bold text-gray-900">

                          {statistics.topCategory.category}

                        </p>

                        <p className="text-sm text-gray-500">

                          {statistics.topCategory.count || 0} transactions

                        </p>

                      </div>

                    </div>

                    <div className="pt-3 border-t border-gray-200">

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

                        <span className="text-sm text-gray-600">Total Spent</span>

                        <span className="text-xl font-bold text-gray-900">

                          {formatCurrency(statistics.topCategory.total || 0)}

                        </span>

                      </div>

                      <div className="flex justify-between items-center mt-2">

                        <span className="text-sm text-gray-600">

                          Percentage of Total

                        </span>

                        <span className="text-lg font-bold text-blue-600">

                          {(statistics.topCategory.percentage || 0).toFixed(1)}%

                        </span>

                      </div>

                    </div>

                  </div>

                ) : (

                  <p className="text-gray-500 text-center py-8">

                    No category data available

                  </p>

                )}

              </div>

              {/\* Most Used Payment \*/}

              <div className="bg-white rounded-xl shadow-lg p-6">

                <h3 className="text-lg font-bold text-gray-900 mb-4">

                  Most Used Payment

                </h3>

                {statistics.mostUsedPayment ? (

                  <div className="space-y-3">

                    <div className="flex items-center gap-3">

                      <div className="w-12 h-12 bg-purple-100 rounded-lg flex items-center justify-center">

                        <MdAccountBalanceWallet className="text-2xl text-purple-600" />

                      </div>

                      <div className="flex-1">

                        <p className="text-xl font-bold text-gray-900">

                          {statistics.mostUsedPayment.paymentmode}

                        </p>

                        <p className="text-sm text-gray-500">

                          {statistics.mostUsedPayment.count || 0} transactions

                        </p>

                      </div>

                    </div>

                    <div className="pt-3 border-t border-gray-200">

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

                        <span className="text-sm text-gray-600">Total Spent</span>

                        <span className="text-xl font-bold text-gray-900">

                          {formatCurrency(statistics.mostUsedPayment.total || 0)}

                        </span>

                      </div>

                      <div className="flex justify-between items-center mt-2">

                        <span className="text-sm text-gray-600">

                          Percentage of Total

                        </span>

                        <span className="text-lg font-bold text-purple-600">

                          {(statistics.mostUsedPayment.percentage || 0).toFixed(1)}%

                        </span>

                      </div>

                    </div>

                  </div>

                ) : (

                  <p className="text-gray-500 text-center py-8">

                    No payment data available

                  </p>

                )}

              </div>

            </div>

            {/\* Activity Summary \*/}

            <div className="bg-white rounded-xl shadow-lg p-6">

              <h3 className="text-lg font-bold text-gray-900 mb-4">

                Account Activity

              </h3>

              <div className="grid grid-cols-1 md:grid-cols-2 gap-4">

                <div className="p-4 bg-blue-50 rounded-lg border border-blue-200">

                  <p className="text-sm text-blue-600 font-medium mb-1">

                    Account Created

                  </p>

                  <p className="text-lg font-bold text-gray-900">

                    {userData?.created\_at

                      ? new Date(userData.created\_at).toLocaleDateString('en-US', {

                          year: 'numeric',

                          month: 'long',

                          day: 'numeric',

                        })

                      : 'N/A'}

                  </p>

                </div>

                <div className="p-4 bg-green-50 rounded-lg border border-green-200">

                  <p className="text-sm text-green-600 font-medium mb-1">

                    Last Activity

                  </p>

                  <p className="text-lg font-bold text-gray-900">

                    {userData?.updated\_at

                      ? new Date(userData.updated\_at).toLocaleDateString('en-US', {

                          year: 'numeric',

                          month: 'long',

                          day: 'numeric',

                        })

                      : 'N/A'}

                  </p>

                </div>

              </div>

            </div>

            {/\* Quick Actions \*/}

            <div className="bg-gradient-to-r from-blue-500 to-purple-600 rounded-xl shadow-lg p-6">

              <div className="flex flex-col md:flex-row items-center justify-between gap-4">

                <div className="text-white">

                  <h3 className="text-xl font-bold">Ready to explore more?</h3>

                  <p className="text-blue-100 text-sm mt-1">

                    Check out your analytics and insights

                  </p>

                </div>

                <div className="flex gap-3">

                  <Link

                    to="/analytics"

                    className="bg-white text-blue-600 px-6 py-3 rounded-lg font-semibold hover:bg-gray-100 transition-colors shadow-lg whitespace-nowrap"

                  >

                    View Analytics

                  </Link>

                  <Link

                    to="/expenses/add"

                    className="bg-blue-700 text-white px-6 py-3 rounded-lg font-semibold hover:bg-blue-800 transition-colors shadow-lg whitespace-nowrap"

                  >

                    Add Expense

                  </Link>

                </div>

              </div>

            </div>

          </div>

        </div>

      </div>

    </div>

  );

};

export default Profile;

E:\expense\_tracker\frontend\src\pages\profile\Settings.jsx

/\*\*

 \* Settings Component

 \* Account settings page for profile updates and password changes

 \* Backend: User model with fullname update capability

 \*/

import { useState } from 'react';

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

import { toast } from 'react-hot-toast';

import {

  MdArrowBack,

  MdPerson,

  MdLock,

  MdSave,

  MdLogout,

  MdWarning,

} from 'react-icons/md';

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

import { getUserData, logout } from '../../utils/auth';

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

const Settings = () => {

  const navigate = useNavigate();

  const userData = getUserData();

  // Profile update state

  const [profileData, setProfileData] = useState({

    fullname: userData?.fullname || '',

    email: userData?.email || '',

  });

  const [profileLoading, setProfileLoading] = useState(false);

  // Password change state

  const [passwordData, setPasswordData] = useState({

    currentPassword: '',

    newPassword: '',

    confirmPassword: '',

  });

  const [passwordLoading, setPasswordLoading] = useState(false);

  const [showPasswords, setShowPasswords] = useState(false);

  // Logout modal

  const [showLogoutModal, setShowLogoutModal] = useState(false);

  const handleProfileChange = (e) => {

    const { name, value } = e.target;

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

  };

  const handlePasswordChange = (e) => {

    const { name, value } = e.target;

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

  };

  const handleProfileUpdate = async (e) => {

    e.preventDefault();

    // Validate fullname

    if (profileData.fullname && profileData.fullname.length > 100) {

      toast.error('Full name is too long (max 100 characters)');

      return;

    }

    setProfileLoading(true);

    try {

      // ✅ FIXED: Changed toast.info() to toast()

      toast('Profile update feature coming soon');

      // In production, you would call:

      // await authAPI.updateProfile({ fullname: profileData.fullname });

      // updateUserData({ ...userData, fullname: profileData.fullname });

      // toast.success('Profile updated successfully');

    } catch (error) {

      console.error('Profile update error:', error);

      toast.error('Failed to update profile');

    } finally {

      setProfileLoading(false);

    }

  };

  const handlePasswordUpdate = async (e) => {

    e.preventDefault();

    // Validate passwords

    if (!passwordData.currentPassword || !passwordData.newPassword) {

      toast.error('Please fill in all password fields');

      return;

    }

    if (passwordData.newPassword.length < 6) {

      toast.error('New password must be at least 6 characters');

      return;

    }

    if (passwordData.newPassword !== passwordData.confirmPassword) {

      toast.error('New passwords do not match');

      return;

    }

    setPasswordLoading(true);

    try {

      // ✅ FIXED: Changed toast.info() to toast()

      toast('Password change feature coming soon');

      // In production, you would call:

      // await authAPI.changePassword({

      //   current\_password: passwordData.currentPassword,

      //   new\_password: passwordData.newPassword,

      // });

      // Clear form

      // setPasswordData({

      //   currentPassword: '',

      //   newPassword: '',

      //   confirmPassword: '',

      // });

      // toast.success('Password changed successfully. Please login again.');

      // handleLogout();

    } catch (error) {

      console.error('Password change error:', error);

      toast.error(error.response?.data?.error || 'Failed to change password');

    } finally {

      setPasswordLoading(false);

    }

  };

  const handleLogout = async () => {

    try {

      await logout();

      toast.success('Logged out successfully');

      navigate('/login');

    } catch (error) {

      console.error('Logout error:', error);

      toast.error('Logout failed');

    }

  };

  return (

    <div className="min-h-screen bg-gray-50 p-6">

      <div className="max-w-4xl mx-auto">

        {/\* Header \*/}

        <div className="mb-6">

          <Link

            to="/profile"

            className="inline-flex items-center gap-2 text-blue-600 hover:text-blue-700 mb-4"

          >

            <MdArrowBack className="text-xl" />

            Back to Profile

          </Link>

          <h1 className="text-3xl font-bold text-gray-900">Account Settings</h1>

          <p className="mt-1 text-sm text-gray-600">

            Manage your account preferences and security

          </p>

        </div>

        <div className="space-y-6">

          {/\* Profile Information \*/}

          <div className="bg-white rounded-xl shadow-lg p-6">

            <div className="flex items-center gap-3 mb-6">

              <div className="w-10 h-10 bg-blue-100 rounded-full flex items-center justify-center">

                <MdPerson className="text-xl text-blue-600" />

              </div>

              <h2 className="text-xl font-bold text-gray-900">

                Profile Information

              </h2>

            </div>

            <form onSubmit={handleProfileUpdate} className="space-y-4">

              {/\* Full Name \*/}

              <div>

                <label

                  htmlFor="fullname"

                  className="block text-sm font-medium text-gray-700 mb-2"

                >

                  Full Name

                </label>

                <input

                  type="text"

                  id="fullname"

                  name="fullname"

                  value={profileData.fullname}

                  onChange={handleProfileChange}

                  maxLength="100"

                  className="w-full px-4 py-3 border border-gray-300 rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-blue-500"

                  placeholder="Enter your full name"

                />

                <p className="text-xs text-gray-500 mt-1">

                  {profileData.fullname.length}/100 characters

                </p>

              </div>

              {/\* Email (Read-only) \*/}

              <div>

                <label

                  htmlFor="email"

                  className="block text-sm font-medium text-gray-700 mb-2"

                >

                  Email Address

                </label>

                <input

                  type="email"

                  id="email"

                  value={profileData.email}

                  disabled

                  className="w-full px-4 py-3 border border-gray-300 rounded-lg bg-gray-100 cursor-not-allowed"

                />

                <p className="text-xs text-gray-500 mt-1">

                  Email cannot be changed

                </p>

              </div>

              {/\* Submit Button \*/}

              <div className="pt-4">

                <button

                  type="submit"

                  disabled={profileLoading}

                  className="flex items-center justify-center gap-2 px-6 py-3 bg-blue-600 text-white rounded-lg font-medium hover:bg-blue-700 disabled:opacity-50 disabled:cursor-not-allowed transition-all"

                >

                  {profileLoading ? (

                    <>

                      <svg

                        className="animate-spin h-5 w-5 text-white"

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

                        fill="none"

                        viewBox="0 0 24 24"

                      >

                        <circle

                          className="opacity-25"

                          cx="12"

                          cy="12"

                          r="10"

                          stroke="currentColor"

                          strokeWidth="4"

                        ></circle>

                        <path

                          className="opacity-75"

                          fill="currentColor"

                          d="M4 12a8 8 0 018-8V0C5.373 0 0 5.373 0 12h4zm2 5.291A7.962 7.962 0 014 12H0c0 3.042 1.135 5.824 3 7.938l3-2.647z"

                        ></path>

                      </svg>

                      Updating...

                    </>

                  ) : (

                    <>

                      <MdSave className="text-xl" />

                      Update Profile

                    </>

                  )}

                </button>

              </div>

            </form>

          </div>

          {/\* Change Password \*/}

          <div className="bg-white rounded-xl shadow-lg p-6">

            <div className="flex items-center gap-3 mb-6">

              <div className="w-10 h-10 bg-purple-100 rounded-full flex items-center justify-center">

                <MdLock className="text-xl text-purple-600" />

              </div>

              <h2 className="text-xl font-bold text-gray-900">Change Password</h2>

            </div>

            <form onSubmit={handlePasswordUpdate} className="space-y-4">

              {/\* Current Password \*/}

              <div>

                <label

                  htmlFor="currentPassword"

                  className="block text-sm font-medium text-gray-700 mb-2"

                >

                  Current Password

                </label>

                <input

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

                  id="currentPassword"

                  name="currentPassword"

                  value={passwordData.currentPassword}

                  onChange={handlePasswordChange}

                  className="w-full px-4 py-3 border border-gray-300 rounded-lg focus:ring-2 focus:ring-purple-500 focus:border-purple-500"

                  placeholder="Enter current password"

                />

              </div>

              {/\* New Password \*/}

              <div>

                <label

                  htmlFor="newPassword"

                  className="block text-sm font-medium text-gray-700 mb-2"

                >

                  New Password

                </label>

                <input

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

                  id="newPassword"

                  name="newPassword"

                  value={passwordData.newPassword}

                  onChange={handlePasswordChange}

                  className="w-full px-4 py-3 border border-gray-300 rounded-lg focus:ring-2 focus:ring-purple-500 focus:border-purple-500"

                  placeholder="Enter new password (min 6 characters)"

                />

              </div>

              {/\* Confirm New Password \*/}

              <div>

                <label

                  htmlFor="confirmPassword"

                  className="block text-sm font-medium text-gray-700 mb-2"

                >

                  Confirm New Password

                </label>

                <input

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

                  id="confirmPassword"

                  name="confirmPassword"

                  value={passwordData.confirmPassword}

                  onChange={handlePasswordChange}

                  className="w-full px-4 py-3 border border-gray-300 rounded-lg focus:ring-2 focus:ring-purple-500 focus:border-purple-500"

                  placeholder="Confirm new password"

                />

              </div>

              {/\* Show Passwords Toggle \*/}

              <div className="flex items-center">

                <input

                  type="checkbox"

                  id="showPasswords"

                  checked={showPasswords}

                  onChange={(e) => setShowPasswords(e.target.checked)}

                  className="w-4 h-4 text-purple-600 border-gray-300 rounded focus:ring-purple-500"

                />

                <label

                  htmlFor="showPasswords"

                  className="ml-2 text-sm text-gray-700"

                >

                  Show passwords

                </label>

              </div>

              {/\* Submit Button \*/}

              <div className="pt-4">

                <button

                  type="submit"

                  disabled={passwordLoading}

                  className="flex items-center justify-center gap-2 px-6 py-3 bg-purple-600 text-white rounded-lg font-medium hover:bg-purple-700 disabled:opacity-50 disabled:cursor-not-allowed transition-all"

                >

                  {passwordLoading ? (

                    <>

                      <svg

                        className="animate-spin h-5 w-5 text-white"

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

                        fill="none"

                        viewBox="0 0 24 24"

                      >

                        <circle

                          className="opacity-25"

                          cx="12"

                          cy="12"

                          r="10"

                          stroke="currentColor"

                          strokeWidth="4"

                        ></circle>

                        <path

                          className="opacity-75"

                          fill="currentColor"

                          d="M4 12a8 8 0 018-8V0C5.373 0 0 5.373 0 12h4zm2 5.291A7.962 7.962 0 014 12H0c0 3.042 1.135 5.824 3 7.938l3-2.647z"

                        ></path>

                      </svg>

                      Updating...

                    </>

                  ) : (

                    <>

                      <MdLock className="text-xl" />

                      Change Password

                    </>

                  )}

                </button>

              </div>

            </form>

          </div>

          {/\* Logout Section \*/}

          <div className="bg-red-50 border border-red-200 rounded-xl shadow-lg p-6">

            <div className="flex items-start gap-3">

              <div className="w-10 h-10 bg-red-100 rounded-full flex items-center justify-center flex-shrink-0">

                <MdWarning className="text-xl text-red-600" />

              </div>

              <div className="flex-1">

                <h3 className="text-lg font-bold text-gray-900 mb-2">

                  Logout from Account

                </h3>

                <p className="text-sm text-gray-600 mb-4">

                  End your current session and return to the login page.

                </p>

                <button

                  onClick={() => setShowLogoutModal(true)}

                  className="flex items-center gap-2 px-6 py-3 bg-red-600 text-white rounded-lg font-medium hover:bg-red-700 transition-all"

                >

                  <MdLogout className="text-xl" />

                  Logout

                </button>

              </div>

            </div>

          </div>

        </div>

      </div>

      {/\* Logout Confirmation Modal \*/}

      {showLogoutModal && (

        <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center z-50 p-4">

          <div className="bg-white rounded-xl shadow-2xl max-w-md w-full p-6">

            <h3 className="text-xl font-bold text-gray-900 mb-2">

              Confirm Logout

            </h3>

            <p className="text-gray-600 mb-6">

              Are you sure you want to logout? You'll need to login again to access

              your account.

            </p>

            <div className="flex gap-3">

              <button

                onClick={() => setShowLogoutModal(false)}

                className="flex-1 px-4 py-2 border border-gray-300 rounded-lg text-gray-700 font-medium hover:bg-gray-50 transition-colors"

              >

                Cancel

              </button>

              <button

                onClick={handleLogout}

                className="flex-1 px-4 py-2 bg-red-600 text-white rounded-lg font-medium hover:bg-red-700 transition-colors"

              >

                Logout

              </button>

            </div>

          </div>

        </div>

      )}

    </div>

  );

};

export default Settings;

E:\expense\_tracker\frontend\src\routes

E:\expense\_tracker\frontend\src\routes\index.jsx

/\*\*

 \* Main Routes Configuration

 \* Defines all application routes with lazy loading for better performance

 \*/

import { lazy, Suspense } from 'react';

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

import ProtectedRoute from './ProtectedRoute';

import PublicRoute from './PublicRoute';

import ROUTES from './routes.config';

// Layouts

import MainLayout from '../layouts/MainLayout';

import AuthLayout from '../layouts/AuthLayout';

// Loading component with better styling

const LoadingSpinner = () => (

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

    <div className="text-center">

      <div className="animate-spin rounded-full h-16 w-16 border-b-4 border-blue-600 mx-auto"></div>

      <p className="mt-4 text-gray-600 font-medium">Loading...</p>

    </div>

  </div>

);

// Lazy load pages for code splitting

// Auth pages

const Login = lazy(() => import('../pages/auth/Login'));

const Register = lazy(() => import('../pages/auth/Register'));

const ForgotPassword = lazy(() => import('../pages/auth/ForgotPassword'));

// Dashboard

const Dashboard = lazy(() => import('../pages/dashboard/Dashboard'));

// Expense pages

const ExpenseList = lazy(() => import('../pages/expenses/ExpenseList'));

const AddExpense = lazy(() => import('../pages/expenses/AddExpense'));

const EditExpense = lazy(() => import('../pages/expenses/EditExpense'));

const ExpenseDetails = lazy(() => import('../pages/expenses/ExpenseDetails'));

// Analytics pages

const Analytics = lazy(() => import('../pages/analytics/Analytics'));

const CategoryBreakdown = lazy(() => import('../pages/analytics/CategoryBreakdown'));

const PaymentAnalysis = lazy(() => import('../pages/analytics/PaymentAnalysis'));

const TrendAnalysis = lazy(() => import('../pages/analytics/TrendAnalysis'));

// AI pages

const AiInsights = lazy(() => import('../pages/ai/AiInsights'));

// Export pages

const ExportReports = lazy(() => import('../pages/export/ExportReports'));

// Profile pages

const Profile = lazy(() => import('../pages/profile/Profile'));

const Settings = lazy(() => import('../pages/profile/Settings'));

// Error pages

const NotFound = lazy(() => import('../pages/NotFound'));

const Unauthorized = lazy(() => import('../pages/Unauthorized'));

/\*\*

 \* Main Router Component

 \* Handles all application routing with authentication protection

 \*/

const AppRoutes = () => {

  return (

    <Suspense fallback={<LoadingSpinner />}>

      <Routes>

        {/\* Root redirect - redirect to login if not authenticated, else dashboard \*/}

        <Route path="/" element={<Navigate to="/dashboard" replace />} />

        {/\* Public routes - Auth Layout \*/}

        <Route element={<AuthLayout />}>

          <Route

            path={ROUTES.LOGIN}

            element={

              <PublicRoute>

                <Login />

              </PublicRoute>

            }

          />

          <Route

            path={ROUTES.REGISTER}

            element={

              <PublicRoute>

                <Register />

              </PublicRoute>

            }

          />

          <Route

            path={ROUTES.FORGOT\_PASSWORD}

            element={

              <PublicRoute>

                <ForgotPassword />

              </PublicRoute>

            }

          />

        </Route>

        {/\* Protected routes - Main Layout \*/}

        <Route

          element={

            <ProtectedRoute>

              <MainLayout />

            </ProtectedRoute>

          }

        >

          {/\* Dashboard \*/}

          <Route path={ROUTES.DASHBOARD} element={<Dashboard />} />

          {/\* Expense Management \*/}

          <Route path={ROUTES.EXPENSES} element={<ExpenseList />} />

          <Route path={ROUTES.ADD\_EXPENSE} element={<AddExpense />} />

          <Route path={ROUTES.EDIT\_EXPENSE} element={<EditExpense />} />

          <Route path={ROUTES.EXPENSE\_DETAILS} element={<ExpenseDetails />} />

          {/\* Analytics \*/}

          <Route path={ROUTES.ANALYTICS} element={<Analytics />} />

          <Route path={ROUTES.CATEGORY\_BREAKDOWN} element={<CategoryBreakdown />} />

          <Route path={ROUTES.PAYMENT\_ANALYSIS} element={<PaymentAnalysis />} />

          <Route path={ROUTES.TREND\_ANALYSIS} element={<TrendAnalysis />} />

          {/\* AI Insights \*/}

          <Route path={ROUTES.AI\_INSIGHTS} element={<AiInsights />} />

          {/\* Export Reports \*/}

          <Route path={ROUTES.EXPORT\_REPORTS} element={<ExportReports />} />

          {/\* Profile & Settings \*/}

          <Route path={ROUTES.PROFILE} element={<Profile />} />

          <Route path={ROUTES.SETTINGS} element={<Settings />} />

        </Route>

        {/\* Error routes - Outside layouts for full-screen display \*/}

        <Route path={ROUTES.UNAUTHORIZED} element={<Unauthorized />} />

        <Route path={ROUTES.NOT\_FOUND} element={<NotFound />} />

        {/\* Catch all route - 404 \*/}

        <Route path="\*" element={<NotFound />} />

      </Routes>

    </Suspense>

  );

};

export default AppRoutes;

E:\expense\_tracker\frontend\src\routes\ProtectedRoute.jsx

/\*\*

 \* ProtectedRoute Component

 \* Wrapper component that protects routes requiring authentication

 \* Redirects to login if user is not authenticated

 \*/

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

import { isAuthenticated } from '../utils/auth';

const ProtectedRoute = ({ children }) => {

  const location = useLocation();

  console.log('🛡️ ProtectedRoute Check');

  console.log('   Current path:', location.pathname);

  const authenticated = isAuthenticated();

  console.log('   Is authenticated:', authenticated);

  // Check if user is authenticated

  if (!authenticated) {

    console.log('⛔ Not authenticated - redirecting to /login');

    console.log('   Saving return path:', location.pathname);

    // Redirect to login page, but save the location they were trying to access

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

  }

  console.log('✅ Authenticated - rendering protected content');

  // User is authenticated, render the protected component

  return children;

};

export default ProtectedRoute;

E:\expense\_tracker\frontend\src\routes\PublicRoute.jsx

/\*\*

 \* PublicRoute Component

 \* Wrapper for routes that should only be accessible when NOT authenticated

 \*/

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

import { isAuthenticated } from '../utils/auth';

const PublicRoute = ({ children }) => {

  const location = useLocation();

  console.log('🌐 PublicRoute Check');

  console.log('   Current path:', location.pathname);

  const authenticated = isAuthenticated();

  console.log('   Is authenticated:', authenticated);

  // If authenticated, redirect to dashboard

  if (authenticated) {

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

    console.log('✅ Already authenticated - redirecting to:', from);

    // IMPORTANT: Don't render children, just redirect

    return <Navigate to={from} replace />;

  }

  console.log('👤 Not authenticated - showing public page');

  return children;

};

export default PublicRoute;

E:\expense\_tracker\frontend\src\routes\routes.config.js

/\*\*

 \* Route Configuration

 \* Defines all route paths and metadata for the application

 \*/

// Route path constants

export const ROUTES = {

  // Public routes

  LOGIN: '/login',

  REGISTER: '/register',

  FORGOT\_PASSWORD: '/forgot-password',

  // Protected routes

  DASHBOARD: '/dashboard',

  // Expense routes

  EXPENSES: '/expenses',

  ADD\_EXPENSE: '/expenses/add',

  EDIT\_EXPENSE: '/expenses/edit/:id',

  EXPENSE\_DETAILS: '/expenses/:id',

  // Analytics routes

  ANALYTICS: '/analytics',

  CATEGORY\_BREAKDOWN: '/analytics/category',

  PAYMENT\_ANALYSIS: '/analytics/payment',

  TREND\_ANALYSIS: '/analytics/trends',

  // AI routes

  AI\_INSIGHTS: '/ai-insights',

  // Export routes

  EXPORT\_REPORTS: '/export',

  // Profile routes

  PROFILE: '/profile',

  SETTINGS: '/settings',

  // Error routes

  NOT\_FOUND: '/404',

  UNAUTHORIZED: '/401',

};

// Route metadata

export const ROUTE\_METADATA = {

  [ROUTES.DASHBOARD]: {

    title: 'Dashboard',

    requiresAuth: true,

    showInNav: true,

    icon: 'MdDashboard',

  },

  [ROUTES.EXPENSES]: {

    title: 'Expenses',

    requiresAuth: true,

    showInNav: true,

    icon: 'MdAccountBalanceWallet',

  },

  [ROUTES.ANALYTICS]: {

    title: 'Analytics',

    requiresAuth: true,

    showInNav: true,

    icon: 'MdBarChart',

  },

  [ROUTES.AI\_INSIGHTS]: {

    title: 'AI Insights',

    requiresAuth: true,

    showInNav: true,

    icon: 'MdPsychology',

  },

  [ROUTES.EXPORT\_REPORTS]: {

    title: 'Export',

    requiresAuth: true,

    showInNav: true,

    icon: 'MdFileDownload',

  },

  [ROUTES.PROFILE]: {

    title: 'Profile',

    requiresAuth: true,

    showInNav: true,

    icon: 'MdPerson',

  },

};

// API endpoints - matching backend routes

export const API\_ENDPOINTS = {

  // Auth endpoints

  AUTH: {

    REGISTER: '/api/auth/register',

    LOGIN: '/api/auth/login',

    LOGOUT: '/api/auth/logout',

    REFRESH: '/api/auth/refresh',

  },

  // Expense endpoints

  EXPENSES: {

    BASE: '/api/expenses',

    BY\_ID: (id) => `/api/expenses/${id}`,

    SUMMARY: '/api/expenses/summary',

  },

  // Category endpoints

  CATEGORIES: {

    BASE: '/api/expenses/categories',

  },

  // Payment mode endpoints

  PAYMENT\_MODES: {

    BASE: '/api/expenses/payment-modes',

  },

  // Analytics endpoints

  ANALYTICS: {

    CATEGORY\_BREAKDOWN: '/api/analytics/category-breakdown',

    PAYMENT\_BREAKDOWN: '/api/analytics/payment-breakdown',

    CROSS\_TAB: '/api/analytics/cross-tab',

    DAILY\_TREND: '/api/analytics/daily-trend',

    MONTHLY\_SUMMARY: '/api/analytics/monthly-summary',

  },

  // AI endpoints

  AI: {

    QUERY: '/api/ai/query',

    SUGGESTIONS: '/api/ai/suggestions',

  },

  // Export endpoints

  EXPORT: {

    CSV: '/api/export/csv',

    PDF: '/api/export/pdf',

    SUMMARY\_PDF: '/api/export/summary-pdf',

  },

};

export default ROUTES;

E:\expense\_tracker\frontend\src\services

E:\expense\_tracker\frontend\src\services\api.js

/\*\*

 \* API Service

 \* Centralized API calls with axios interceptors for authentication

 \*/

import axios from 'axios';

import { getAuthToken, logout } from '../utils/auth';

import { toast } from 'react-hot-toast';

// Base API URL

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

console.log('🌐 API Base URL:', API\_BASE\_URL);

// Create axios instance

const apiClient = axios.create({

  baseURL: API\_BASE\_URL,

  headers: {

    'Content-Type': 'application/json',

  },

  timeout: 30000,

  withCredentials: false,

});

// Request interceptor

apiClient.interceptors.request.use(

  (config) => {

    console.log('📤 API Request:', {

      method: config.method?.toUpperCase(),

      url: config.url,

      fullURL: `${config.baseURL}${config.url}`,

    });

    const token = getAuthToken();

    if (token) {

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

      console.log('🔐 Token attached');

    } else {

      console.log('⚠️ No token');

    }

    return config;

  },

  (error) => {

    console.error('❌ Request error:', error);

    return Promise.reject(error);

  }

);

// ✅ IMPROVED: Response interceptor with better error logging

apiClient.interceptors.response.use(

  (response) => {

    console.log('✅ Response:', {

      status: response.status,

      url: response.config.url,

    });

    // ✅ ADDED: Log auth responses for debugging

    if (response.config.url.includes('/auth/login') || response.config.url.includes('/auth/register')) {

      console.log('🔍 AUTH RESPONSE DATA:', response.data);

      console.log('🔍 USER OBJECT:', response.data.user);

      console.log('🔍 FULL\_NAME VALUE:', response.data.user?.full\_name);

    }

    return response;

  },

  async (error) => {

    // ✅ IMPROVED: Better error logging to see backend errors

    console.error('❌ API Error:', {

      message: error.message,

      status: error.response?.status,

      url: error.config?.url,

      // ✅ ADDED: Show full backend error response

      backendError: error.response?.data?.error || error.response?.data?.message,

      fullErrorData: error.response?.data,

    });

    // ✅ ADDED: Log full error for 500 errors

    if (error.response?.status === 500) {

      console.error('🔥 BACKEND 500 ERROR DETAILS:', {

        url: error.config?.url,

        method: error.config?.method,

        params: error.config?.params,

        errorMessage: error.response?.data?.error,

        fullResponse: error.response?.data,

      });

    }

    const originalRequest = error.config;

    if (!error.response) {

      toast.error('Cannot connect to server');

      return Promise.reject(error);

    }

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

      const url = originalRequest.url;

      const isAuth = url.includes('/auth/');

      const isUser = url.includes('/user/');

      if ((isAuth || isUser) && !originalRequest.\_retry) {

        originalRequest.\_retry = true;

        console.log('🔒 Auth failed on auth/user route - logging out');

        await logout();

        window.location.href = '/login';

        toast.error('Session expired');

        return Promise.reject(error);

      }

      console.warn('⚠️ 401 on non-auth route:', url);

    }

    if (error.response.status === 500) {

      // ✅ IMPROVED: Show backend error message if available

      const backendMessage = error.response?.data?.error || error.response?.data?.message;

      toast.error(backendMessage || 'Server error occurred');

    }

    return Promise.reject(error);

  }

);

// ============================================================================

// AUTH API

// ============================================================================

export const authAPI = {

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

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

  logout: () => apiClient.post('/auth/logout'),

};

// ============================================================================

// EXPENSE API

// ============================================================================

export const expenseAPI = {

  getExpenses: (params) => apiClient.get('/expenses', { params }),

  getExpenseById: (id) => apiClient.get(`/expenses/${id}`),

  createExpense: (data) => apiClient.post('/expenses/add', data), // fixed

  updateExpense: (id, data) => apiClient.put(`/expenses/${id}`, data),

  deleteExpense: (id) => apiClient.delete(`/expenses/${id}`),

  getCategories: () => apiClient.get('/expenses/categories'),

  getPaymentModes: () => apiClient.get('/expenses/payment-modes'),

  getSummary: (params) => apiClient.get('/expenses/summary', { params }),

};

// ============================================================================

// ANALYTICS API

// ============================================================================

export const analyticsAPI = {

  getCategoryBreakdown: (startDate, endDate) =>

    apiClient.get('/analytics/categories-vs-expenses', {

      params: { start\_date: startDate, end\_date: endDate },

    }),

  getPaymentBreakdown: (startDate, endDate) =>

    apiClient.get('/analytics/payment-modes-vs-expenses', {

      params: { start\_date: startDate, end\_date: endDate },

    }),

  getCrossTab: (startDate, endDate) =>

    apiClient.get('/analytics/payment-modes-vs-categories', {

      params: { start\_date: startDate, end\_date: endDate },

    }),

  getDailyTrend: (days = 7) =>

    apiClient.get('/analytics/daily-trend', {

      params: { days },

    }),

  getMonthlySummary: () =>

    apiClient.get('/analytics/monthly-summary'),

};

// ============================================================================

// AI API

// ============================================================================

export const aiAPI = {

  query: (data) => apiClient.post('/ai/query', data),

  getSuggestions: () => apiClient.get('/ai/suggestions'),

};

// ============================================================================

// EXPORT API

// ============================================================================

export const exportAPI = {

  exportCSV: (params) =>

    apiClient.get(`/export/csv?${params}`, {

      responseType: 'blob',

    }),

  exportPDF: (params) =>

    apiClient.get(`/export/pdf?${params}`, {

      responseType: 'blob',

    }),

};

// ============================================================================

// USER API

// ============================================================================

export const userAPI = {

  getProfile: () => apiClient.get('/user/profile'),

  updateProfile: (data) => apiClient.put('/user/profile', data),

  changePassword: (data) => apiClient.post('/user/change-password', data),

};

export default apiClient;

E:\expense\_tracker\frontend\src\utils

E:\expense\_tracker\frontend\src\utils\auth.js

/\*\*

 \* Authentication Utility Functions

 \* Handles JWT token storage, validation, and user session management

 \*/

// Token and user data keys for localStorage

const TOKEN\_KEY = 'expense\_tracker\_token';

const USER\_KEY = 'expense\_tracker\_user';

/\*\*

 \* Check if user is authenticated

 \* @returns {boolean} - True if valid token exists

 \*/

export const isAuthenticated = () => {

  console.log('🔍 === AUTHENTICATION CHECK START ===');

  try {

    const token = getAuthToken();

    console.log('1️⃣ Token exists:', !!token);

    console.log('   Token value:', token ? token.substring(0, 50) + '...' : 'null');

    // If no token exists, user is not authenticated

    if (!token) {

      console.log('❌ RESULT: Not authenticated (no token)');

      console.log('🔍 === AUTHENTICATION CHECK END ===\n');

      return false;

    }

    // Validate token format (JWT has 3 parts separated by dots)

    const parts = token.split('.');

    console.log('2️⃣ Token parts count:', parts.length);

    if (parts.length !== 3) {

      console.warn('⚠️ Invalid token format - expected 3 parts, got', parts.length);

      removeAuthToken(); // Clear invalid token

      console.log('❌ RESULT: Not authenticated (invalid format)');

      console.log('🔍 === AUTHENTICATION CHECK END ===\n');

      return false;

    }

    // Check if token is expired

    try {

      const payload = JSON.parse(atob(parts[1]));

      console.log('3️⃣ Token payload decoded:', payload);

      // Check if payload has expiration

      if (!payload.exp) {

        console.log('⚠️ Token has no expiration - allowing access');

        console.log('✅ RESULT: Authenticated (no expiration check)');

        console.log('🔍 === AUTHENTICATION CHECK END ===\n');

        return true; // Allow if no expiration set

      }

      const expirationTime = payload.exp \* 1000; // Convert to milliseconds

      const currentTime = Date.now();

      const isValid = currentTime < expirationTime;

      console.log('4️⃣ Token expiration check:');

      console.log('   Expires at:', new Date(expirationTime).toLocaleString());

      console.log('   Current time:', new Date(currentTime).toLocaleString());

      console.log('   Is valid:', isValid);

      console.log('   Time remaining:', Math.round((expirationTime - currentTime) / 1000 / 60), 'minutes');

      // Clear token if expired

      if (!isValid) {

        console.log('❌ Token has expired');

        logout();

        console.log('❌ RESULT: Not authenticated (expired)');

        console.log('🔍 === AUTHENTICATION CHECK END ===\n');

        return false;

      }

      console.log('✅ RESULT: Authenticated (valid token)');

      console.log('🔍 === AUTHENTICATION CHECK END ===\n');

      return true;

    } catch (decodeError) {

      console.error('❌ Token decode error:', decodeError);

      removeAuthToken(); // Clear invalid token

      console.log('❌ RESULT: Not authenticated (decode error)');

      console.log('🔍 === AUTHENTICATION CHECK END ===\n');

      return false;

    }

  } catch (error) {

    console.error('❌ Authentication check error:', error);

    console.log('❌ RESULT: Not authenticated (exception)');

    console.log('🔍 === AUTHENTICATION CHECK END ===\n');

    return false;

  }

};

/\*\*

 \* Get authentication token from localStorage

 \* @returns {string|null} - JWT token or null

 \*/

export const getAuthToken = () => {

  try {

    return localStorage.getItem(TOKEN\_KEY);

  } catch (error) {

    console.error('Error getting token:', error);

    return null;

  }

};

/\*\*

 \* Set authentication token in localStorage

 \* @param {string} token - JWT token

 \*/

export const setAuthToken = (token) => {

  try {

    if (token) {

      localStorage.setItem(TOKEN\_KEY, token);

      console.log('💾 Token saved to localStorage');

      console.log('   Key:', TOKEN\_KEY);

      console.log('   Value:', token.substring(0, 50) + '...');

    }

  } catch (error) {

    console.error('Error setting token:', error);

  }

};

/\*\*

 \* Remove authentication token from localStorage

 \*/

export const removeAuthToken = () => {

  try {

    localStorage.removeItem(TOKEN\_KEY);

    console.log('🗑️ Token removed from localStorage');

  } catch (error) {

    console.error('Error removing token:', error);

  }

};

/\*\*

 \* Get user data from localStorage

 \* @returns {object|null} - User object or null

 \*/

export const getUserData = () => {

  try {

    const userData = localStorage.getItem(USER\_KEY);

    return userData ? JSON.parse(userData) : null;

  } catch (error) {

    console.error('Error parsing user data:', error);

    return null;

  }

};

/\*\*

 \* Set user data in localStorage

 \* @param {object} user - User object

 \*/

export const setUserData = (user) => {

  try {

    if (user) {

      localStorage.setItem(USER\_KEY, JSON.stringify(user));

      console.log('💾 User data saved to localStorage');

      console.log('   Key:', USER\_KEY);

      console.log('   Value:', user);

    }

  } catch (error) {

    console.error('Error setting user data:', error);

  }

};

/\*\*

 \* Update user data in localStorage

 \* @param {object} updates - Partial user object with updates

 \*/

export const updateUserData = (updates) => {

  try {

    const currentUser = getUserData();

    if (currentUser) {

      const updatedUser = { ...currentUser, ...updates };

      setUserData(updatedUser);

    }

  } catch (error) {

    console.error('Error updating user data:', error);

  }

};

/\*\*

 \* Remove user data from localStorage

 \*/

export const removeUserData = () => {

  try {

    localStorage.removeItem(USER\_KEY);

    console.log('🗑️ User data removed from localStorage');

  } catch (error) {

    console.error('Error removing user data:', error);

  }

};

/\*\*

 \* Logout user - Clear all auth data

 \*/

export const logout = () => {

  console.log('👋 Logging out user...');

  try {

    removeAuthToken();

    removeUserData();

    // Clear any other user-specific data

    localStorage.removeItem('exportHistory');

    console.log('✅ Logout complete');

  } catch (error) {

    console.error('Error during logout:', error);

  }

  return Promise.resolve();

};

/\*\*

 \* Initialize auth - Check token validity on app load

 \* @returns {boolean} - True if session is valid

 \*/

export const initializeAuth = () => {

  if (!isAuthenticated()) {

    logout(); // Clear invalid session

    return false;

  }

  return true;

};

/\*\*

 \* Get authorization header for API requests

 \* @returns {object} - Authorization header object

 \*/

export const getAuthHeader = () => {

  const token = getAuthToken();

  return token ? { Authorization: `Bearer ${token}` } : {};

};

/\*\*

 \* Decode JWT token payload

 \* @param {string} token - JWT token

 \* @returns {object|null} - Decoded payload or null

 \*/

export const decodeToken = (token) => {

  try {

    if (!token) return null;

    const parts = token.split('.');

    if (parts.length !== 3) {

      console.warn('Invalid token format');

      return null;

    }

    const base64Url = parts[1];

    const base64 = base64Url.replace(/-/g, '+').replace(/\_/g, '/');

    const jsonPayload = decodeURIComponent(

      atob(base64)

        .split('')

        .map((c) => '%' + ('00' + c.charCodeAt(0).toString(16)).slice(-2))

        .join('')

    );

    return JSON.parse(jsonPayload);

  } catch (error) {

    console.error('Error decoding token:', error);

    return null;

  }

};

/\*\*

 \* Get user ID from token

 \* @returns {string|null} - User ID or null

 \*/

export const getUserId = () => {

  const token = getAuthToken();

  if (!token) return null;

  const payload = decodeToken(token);

  return payload?.sub || payload?.user\_id || payload?.id || null;

};

/\*\*

 \* Check if token will expire soon (within 5 minutes)

 \* @returns {boolean} - True if token expires soon

 \*/

export const isTokenExpiringSoon = () => {

  try {

    const token = getAuthToken();

    if (!token) return false;

    const payload = decodeToken(token);

    if (!payload || !payload.exp) return false;

    const expirationTime = payload.exp \* 1000;

    const fiveMinutes = 5 \* 60 \* 1000;

    return Date.now() > expirationTime - fiveMinutes;

  } catch (error) {

    console.error('Error checking token expiration:', error);

    return false;

  }

};

/\*\*

 \* Get token expiration time

 \* @returns {Date|null} - Expiration date or null

 \*/

export const getTokenExpiration = () => {

  try {

    const token = getAuthToken();

    if (!token) return null;

    const payload = decodeToken(token);

    if (!payload || !payload.exp) return null;

    return new Date(payload.exp \* 1000);

  } catch (error) {

    console.error('Error getting token expiration:', error);

    return null;

  }

};

/\*\*

 \* Save login data (token and user info)

 \* @param {string} token - JWT token

 \* @param {object} user - User data object

 \*/

export const saveLoginData = (token, user) => {

  console.log('💾 Saving login data...');

  try {

    setAuthToken(token);

    setUserData(user);

    console.log('✅ Login data saved successfully');

  } catch (error) {

    console.error('Error saving login data:', error);

  }

};

E:\expense\_tracker\frontend\src\utils\helpers.js

/\*\*

 \* Helper Utility Functions

 \* Common helper functions used across the application

 \*/

/\*\*

 \* Format currency value

 \* @param {number} amount - Amount to format

 \* @param {string} currency - Currency code (default: INR)

 \* @returns {string} - Formatted currency string

 \*/

export const formatCurrency = (amount, currency = 'INR') => {

  if (amount === null || amount === undefined) return '₹0.00';

  return new Intl.NumberFormat('en-IN', {

    style: 'currency',

    currency: currency,

    minimumFractionDigits: 2,

    maximumFractionDigits: 2,

  }).format(amount);

};

/\*\*

 \* Format date string

 \* @param {string} dateString - ISO date string

 \* @param {string} format - Format type ('short', 'long', 'time')

 \* @returns {string} - Formatted date string

 \*/

export const formatDate = (dateString, format = 'short') => {

  if (!dateString) return 'N/A';

  const date = new Date(dateString);

  if (isNaN(date.getTime())) return 'Invalid Date';

  const options = {

    short: { year: 'numeric', month: 'short', day: 'numeric' },

    long: { year: 'numeric', month: 'long', day: 'numeric' },

    time: {

      year: 'numeric',

      month: 'short',

      day: 'numeric',

      hour: '2-digit',

      minute: '2-digit',

    },

  };

  return date.toLocaleDateString('en-IN', options[format] || options.short);

};

/\*\*

 \* Truncate text to specified length

 \* @param {string} text - Text to truncate

 \* @param {number} maxLength - Maximum length

 \* @returns {string} - Truncated text

 \*/

export const truncateText = (text, maxLength = 50) => {

  if (!text) return '';

  return text.length > maxLength ? text.substring(0, maxLength) + '...' : text;

};

/\*\*

 \* Capitalize first letter of string

 \* @param {string} str - String to capitalize

 \* @returns {string} - Capitalized string

 \*/

export const capitalizeFirst = (str) => {

  if (!str) return '';

  return str.charAt(0).toUpperCase() + str.slice(1);

};

/\*\*

 \* Parse date string to ISO format

 \* @param {string} dateString - Date string

 \* @returns {string} - ISO date string

 \*/

export const parseDate = (dateString) => {

  if (!dateString) return null;

  const date = new Date(dateString);

  return isNaN(date.getTime()) ? null : date.toISOString();

};

/\*\*

 \* Generate random color

 \* @returns {string} - Hex color code

 \*/

export const generateRandomColor = () => {

  return '#' + Math.floor(Math.random() \* 16777215).toString(16);

};

/\*\*

 \* Debounce function

 \* @param {Function} func - Function to debounce

 \* @param {number} wait - Wait time in milliseconds

 \* @returns {Function} - Debounced function

 \*/

export const debounce = (func, wait = 300) => {

  let timeout;

  return function executedFunction(...args) {

    const later = () => {

      clearTimeout(timeout);

      func(...args);

    };

    clearTimeout(timeout);

    timeout = setTimeout(later, wait);

  };

};

E:\expense\_tracker\frontend\src\App.css

.App {

  text-align: center;

}

.App-logo {

  height: 40vmin;

  pointer-events: none;

}

@media (prefers-reduced-motion: no-preference) {

  .App-logo {

    animation: App-logo-spin infinite 20s linear;

  }

}

.App-header {

  background-color: #282c34;

  min-height: 100vh;

  display: flex;

  flex-direction: column;

  align-items: center;

  justify-content: center;

  font-size: calc(10px + 2vmin);

  color: white;

}

.App-link {

  color: #61dafb;

}

@keyframes App-logo-spin {

  from {

    transform: rotate(0deg);

  }

  to {

    transform: rotate(360deg);

  }

}

E:\expense\_tracker\frontend\src\App.js

/\*\*

 \* App Component

 \* Root component that sets up routing and global providers

 \*/

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

import { Toaster } from 'react-hot-toast';

import Router from './router'; // Using router.jsx

import './App.css';

function App() {

  console.log('🚀 App component loaded');

  return (

    <BrowserRouter>

      {/\* Global Toast Notifications \*/}

      <Toaster

        position="top-right"

        reverseOrder={false}

        toastOptions={{

          // Default options

          duration: 4000,

          style: {

            background: '#fff',

            color: '#363636',

            fontSize: '14px',

            fontWeight: '500',

            padding: '16px',

            borderRadius: '8px',

            boxShadow: '0 4px 12px rgba(0, 0, 0, 0.15)',

          },

          // Success

          success: {

            duration: 3000,

            iconTheme: {

              primary: '#10b981',

              secondary: '#fff',

            },

          },

          // Error

          error: {

            duration: 4000,

            iconTheme: {

              primary: '#ef4444',

              secondary: '#fff',

            },

          },

          // Loading

          loading: {

            iconTheme: {

              primary: '#3b82f6',

              secondary: '#fff',

            },

          },

        }}

      />

      {/\* Main Application Routes \*/}

      <Router />

    </BrowserRouter>

  );

}

export default App;

E:\expense\_tracker\frontend\src\index.css

@tailwind base;

@tailwind components;

@tailwind utilities;

E:\expense\_tracker\frontend\src\index.js

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>

);

E:\expense\_tracker\frontend\src\router.jsx

/\*\*

 \* Router Configuration

 \* Alternative router setup file

 \*/

import { lazy, Suspense } from 'react';

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

import ProtectedRoute from './routes/ProtectedRoute';

// Layouts

import MainLayout from './pages/layouts/MainLayout';

import AuthLayout from './pages/layouts/AuthLayout';

// Loading component

const LoadingSpinner = () => (

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

    <div className="text-center">

      <div className="animate-spin rounded-full h-16 w-16 border-b-4 border-blue-600 mx-auto"></div>

      <p className="mt-4 text-gray-600 font-medium">Loading...</p>

    </div>

  </div>

);

// Lazy loaded components

const Login = lazy(() => import('./pages/auth/Login'));

const Register = lazy(() => import('./pages/auth/Register'));

const ForgotPassword = lazy(() => import('./pages/auth/ForgotPassword'));

const Dashboard = lazy(() => import('./pages/dashboard/Dashboard'));

const ExpenseList = lazy(() => import('./pages/expenses/ExpenseList'));

const AddExpense = lazy(() => import('./pages/expenses/AddExpense'));

const EditExpense = lazy(() => import('./pages/expenses/EditExpense'));

const ExpenseDetails = lazy(() => import('./pages/expenses/ExpenseDetails'));

const Analytics = lazy(() => import('./pages/analytics/Analytics'));

const CategoryBreakdown = lazy(() => import('./pages/analytics/CategoryBreakdown'));

const PaymentAnalysis = lazy(() => import('./pages/analytics/PaymentAnalysis'));

const TrendAnalysis = lazy(() => import('./pages/analytics/TrendAnalysis'));

const AiInsights = lazy(() => import('./pages/ai/AiInsights'));

const ExportReports = lazy(() => import('./pages/export/ExportReports'));

const Profile = lazy(() => import('./pages/profile/Profile'));

const Settings = lazy(() => import('./pages/profile/Settings'));

const NotFound = lazy(() => import('./pages/NotFound'));

const Unauthorized = lazy(() => import('./pages/Unauthorized'));

const Router = () => {

  console.log('🔀 Router component rendering');

  return (

    <Suspense fallback={<LoadingSpinner />}>

      <Routes>

        {/\* Root redirect \*/}

        <Route path="/" element={<Navigate to="/dashboard" replace />} />

        {/\* ====================================== \*/}

        {/\* PUBLIC ROUTES (NO PublicRoute wrapper) \*/}

        {/\* ====================================== \*/}

        <Route element={<AuthLayout />}>

          <Route path="/login" element={<Login />} />

          <Route path="/register" element={<Register />} />

          <Route path="/forgot-password" element={<ForgotPassword />} />

        </Route>

        {/\* ====================================== \*/}

        {/\* PROTECTED ROUTES \*/}

        {/\* ====================================== \*/}

        <Route

          element={

            <ProtectedRoute>

              <MainLayout />

            </ProtectedRoute>

          }

        >

          <Route path="/dashboard" element={<Dashboard />} />

          {/\* Expenses \*/}

          <Route path="/expenses" element={<ExpenseList />} />

          <Route path="/expenses/add" element={<AddExpense />} />

          <Route path="/expenses/edit/:id" element={<EditExpense />} />

          <Route path="/expenses/:id" element={<ExpenseDetails />} />

          {/\* Analytics \*/}

          <Route path="/analytics" element={<Analytics />} />

          <Route path="/analytics/category" element={<CategoryBreakdown />} />

          <Route path="/analytics/payment" element={<PaymentAnalysis />} />

          <Route path="/analytics/trends" element={<TrendAnalysis />} />

          {/\* AI & Export \*/}

          <Route path="/ai-insights" element={<AiInsights />} />

          <Route path="/export" element={<ExportReports />} />

          {/\* Profile \*/}

          <Route path="/profile" element={<Profile />} />

          <Route path="/settings" element={<Settings />} />

        </Route>

        {/\* Error routes \*/}

        <Route path="/401" element={<Unauthorized />} />

        <Route path="/404" element={<NotFound />} />

        <Route path="\*" element={<NotFound />} />

      </Routes>

    </Suspense>

  );

};

export default Router;

E:\expense\_tracker\frontend\.env

# Backend API URL

REACT\_APP\_API\_URL=http://localhost:5000/api

# App name

REACT\_APP\_NAME=Expense Tracker

# Environment

REACT\_APP\_ENV=development

E:\expense\_tracker\frontend\.gitignore

# See https://help.github.com/articles/ignoring-files/ for more about ignoring files.

# dependencies

/node\_modules

/.pnp

.pnp.js

# testing

/coverage

# production

/build

# misc

.DS\_Store

.env.local

.env.development.local

.env.test.local

.env.production.local

npm-debug.log\*

yarn-debug.log\*

yarn-error.log\*

E:\expense\_tracker\frontend\package.json

{

  "name": "frontend",

  "version": "0.1.0",

  "private": true,

  "dependencies": {

    "@headlessui/react": "^2.2.9",

    "@testing-library/dom": "^10.4.1",

    "@testing-library/jest-dom": "^6.9.1",

    "@testing-library/react": "^16.3.0",

    "@testing-library/user-event": "^13.5.0",

    "axios": "^1.12.2",

    "date-fns": "^4.1.0",

    "jwt-decode": "^4.0.0",

    "react": "^19.2.0",

    "react-dom": "^19.2.0",

    "react-hot-toast": "^2.6.0",

    "react-icons": "^5.5.0",

    "react-router-dom": "^7.9.4",

    "react-scripts": "5.0.1",

    "recharts": "^3.2.1",

    "web-vitals": "^2.1.4"

  },

  "scripts": {

    "start": "react-scripts start",

    "build": "react-scripts build",

    "test": "react-scripts test",

    "eject": "react-scripts eject"

  },

  "eslintConfig": {

    "extends": [

      "react-app",

      "react-app/jest"

    ]

  },

  "browserslist": {

    "production": [

      ">0.2%",

      "not dead",

      "not op\_mini all"

    ],

    "development": [

      "last 1 chrome version",

      "last 1 firefox version",

      "last 1 safari version"

    ]

  },

  "devDependencies": {

    "autoprefixer": "^10.4.21",

    "postcss": "^8.5.6",

    "tailwindcss": "^3.4.18"

  }

}

E:\expense\_tracker\frontend\tailwind.config.js

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

module.exports = {

  content: [

    "./src/\*\*/\*.{js,jsx,ts,tsx}",

    "./public/index.html"

  ],

  theme: {

    extend: {

      colors: {

        primary: {

          50: '#f0f9ff',

          100: '#e0f2fe',

          200: '#bae6fd',

          300: '#7dd3fc',

          400: '#38bdf8',

          500: '#0ea5e9',

          600: '#0284c7',

          700: '#0369a1',

          800: '#075985',

          900: '#0c4a6e',

        },

      },

    },

  },

  plugins: [],

}