```
import React, { useState, useEffect } from 'react';
import { PlusCircle, Trash2, TrendingUp, TrendingDown, Target, DollarSign, PieChart, BarChart3,
Lightbulb, Eye, EyeOff } from 'lucide-react';
import { PieChart as RechartsPieChart, Cell, ResponsiveContainer, BarChart, Bar, XAxis, YAxis,
CartesianGrid, Tooltip, Legend, Pie } from 'recharts';
const FinancialTracker = () => {
// State management
const [assets, setAssets] = useState([]);
const [liabilities, setLiabilities] = useState([]);
const [funds, setFunds] = useState([]);
const [goals, setGoals] = useState([]);
const [portfolio, setPortfolio] = useState([]);
const [transactions, setTransactions] = useState([]);
const [activeTab, setActiveTab] = useState('overview');
const [showValues, setShowValues] = useState(true);
// Form states
const [assetForm, setAssetForm] = useState({ name: '', value: '', type: 'Real Estate' });
const [liabilityForm, setLiabilityForm] = useState({ name: '', amount: '', type: 'Loan' });
const [fundForm, setFundForm] = useState({ name: '', balance: '', target: '' });
const [goalForm, setGoalForm] = useState({ name: '', target: '', current: '' });
const [portfolioForm, setPortfolioForm] = useState({ symbol: '', shares: '', avgPrice: '', currentPrice: ''
});
// Add transaction helper
const addTransaction = (type, action, item, amount = null) => {
const transaction = {
id: Date.now() + Math.random(),
date: new Date().toISOString(),
type,
action,
item: item.name || item.symbol,
amount: amount || item.value || item.amount || item.balance,
timestamp: Date.now()
};
setTransactions(prev => [transaction, ...prev]);
};
// Asset management
const addAsset = () => {
if ( assetForm.name && assetForm.value) {
const newAsset = {
id: Date.now(),
...assetForm,
value: parseFloat(assetForm.value)
};
setAssets(prev => [...prev, newAsset]);
addTransaction('Asset', 'Added', newAsset);
setAssetForm({ name: '', value: '', type: 'Real Estate' });
}
};
const removeAsset = (id) => {
const asset = assets.find(a => a.id === id);
setAssets(prev => prev.filter(a => a.id !== id));
addTransaction('Asset', 'Removed', asset);
};
// Liability management
const addLiability = () => {
if ( liabilityForm.name && liabilityForm.amount) {
const newLiability = {
```

```
id: Date.now(),
...liabilityForm,
amount: parseFloat(liabilityForm.amount)
};
setLiabilities(prev => [...prev, newLiability]);
addTransaction('Liability', 'Added', newLiability);
setLiabilityForm({ name: '', amount: '', type: 'Loan' });
}
};
const removeLiability = (id) => {
const liability = liabilities.find(l => l.id === id);
setLiabilities(prev => prev.filter(l => l.id !== id));
addTransaction('Liability', 'Removed', liability);
};
// Fund management
const addFund = () => {
if ( fundForm.name && fundForm.balance) {
const newFund = {
id: Date.now(),
...fundForm,
balance: parseFloat(fundForm.balance),
target: parseFloat( fundForm.target) || 0
};
setFunds(prev => [...prev, newFund]);
addTransaction('Fund', 'Created', newFund);
setFundForm({ name: '', balance: '', target: '' });
}
};
const contributeFund = (id, amount) => {
const contribution = parseFloat(amount);
if (contribution > 0) {
setFunds(prev =>
prev.map(fund =>
fund.id === id
? { ...fund, balance: fund.balance + contribution }
: fund
)
);
const fund = funds.find(f => f.id === id);
addTransaction('Fund', 'Contribution', fund, contribution);
}
};
const removeFund = (id) => {
const fund = funds.find(f => f.id === id);
setFunds(prev => prev.filter(f => f.id !== id));
addTransaction('Fund', 'Deleted', fund);
};
// Goal management
const addGoal = () => {
if ( goalForm.name && goalForm.target) {
const newGoal = {
id: Date.now(),
...goalForm,
target: parseFloat( goalForm.target),
current: parseFloat(goalForm.current) || 0
};
setGoals(prev => [...prev, newGoal]);
```

```
addTransaction('Goal', 'Created', newGoal);
setGoalForm({ name: '', target: '', current: '' });
}
};
const updateGoal = (id, amount) => {
const update = parseFloat(amount);
if (update > 0) {
setGoals(prev =>
prev.map(goal =>
goal.id === id
? { ...goal, current: goal.current + update }
: goal
)
);
const goal = goals.find(g => g.id === id);
addTransaction('Goal', 'Progress Update', goal, update);
}
};
const removeGoal = (id) => {
const goal = goals.find(g => g.id === id);
setGoals(prev => prev.filter(g => g.id !== id));
addTransaction('Goal', 'Removed', goal);
};
// Portfolio management
const addStock = () => {
if (portfolioForm.symbol && portfolioForm.shares && portfolioForm.avgPrice) {
const newStock = {
id: Date.now(),
...portfolioForm,
shares: parseFloat(portfolioForm.shares),
avgPrice: parseFloat(portfolioForm.avgPrice),
currentPrice: parseFloat(portfolioForm.currentPrice) || parseFloat(portfolioForm.avgPrice)
};
setPortfolio(prev => [...prev, newStock]);
addTransaction('Portfolio', 'Added Stock', newStock);
setPortfolioForm({ symbol: '', shares: '', avgPrice: '', currentPrice: '' });
}
};
const removeStock = (id) => {
const stock = portfolio.find(s => s.id === id);
setPortfolio(prev => prev.filter(s => s.id !== id));
addTransaction('Portfolio', 'Removed Stock', stock);
};
// Calculations
const totalAssets = assets.reduce((sum, asset) => sum + asset.value, 0);
const totalLiabilities = liabilities.reduce((sum, liability) => sum + liability.amount, 0);
const portfolioValue = portfolio.reduce((sum, stock) => sum + (stock.shares * stock.currentPrice),
0);
const netWorth = totalAssets + portfolioValue - totalLiabilities;
// Chart data preparation
const getAssetsByCategory = () => {
const assetCategories = {};
assets.forEach(asset => {
assetCategories[asset.type] = (assetCategories[asset.type] || 0) + asset.value;
});
if (portfolioValue > 0) {
assetCategories['Portfolio'] = portfolioValue;
```

```
}
return Object.entries(assetCategories).map(([name, value]) => ({ name, value }));
};
const getLiabilitiesByCategory = () => {
const liabilityCategories = {};
liabilities.forEach(liability => {
liabilityCategories[liability.type] = (liabilityCategories[liability.type] || 0) + liability.amount;
});
return Object.entries(liabilityCategories).map(([name, value]) => ({ name, value }));
};
const getNetWorthData = () => [
{ name: 'Assets', value: totalAssets + portfolioValue, color: '#10B981' },
{ name: 'Liabilities', value: totalLiabilities, color: '#EF4444' }
];
const getPortfolioData = () => {
return portfolio.map(stock => ({
symbol: stock.symbol,
value: stock.shares * stock.currentPrice,
shares: stock.shares,
gainLoss: (stock.shares *stock.currentPrice) - (stock.shares* stock.avgPrice),
gainLossPercent: ((stock.currentPrice - stock.avgPrice) / stock.avgPrice) * 100
}));
};
// Chart colors
const COLORS = ['#3B82F6', '#10B981', '#F59E0B', '#EF4444', '#8B5CF6', '#06B6D4', '#84CC16',
'#F97316'];
const CustomTooltip = ({ active, payload, label }) => {
if (active && payload && payload.length) {
return (
<div className="bg-white p-3 border rounded-lg shadow-lg">
<p className="font-medium">{`${label}`}</p>
<p className="text-blue-600">
{`Value: ${formatCurrency(payload[0].value)}`}
</p>
</div>
);
}
return null;
};
// Financial suggestions
const getFinancialSuggestions = () => {
const suggestions = [];

if (netWorth < 0) {
suggestions.push("Your net worth is negative. Focus on paying down high-interest debt first.");
}

const emergencyFund = funds.find(f => f.name.toLowerCase().includes('emergency'));
if (!emergencyFund) {
suggestions.push("Consider creating an emergency fund with 3-6 months of expenses.");
} else if (emergencyFund.balance < 10000) {
suggestions.push("Try to build your emergency fund to at least $10,000.");
}

if (totalLiabilities > totalAssets * 0.3) {
suggestions.push("Your debt-to-asset ratio is high. Consider debt consolidation or aggressive
repayment.");
```

```
}

if (portfolio.length === 0) {
suggestions.push("Consider starting an investment portfolio for long-term wealth building.");
}

const completedGoals = goals.filter(g => g.current >= g.target).length;
if (completedGoals > 0) {
suggestions.push(`Congratulations! You've achieved ${completedGoals} financial goal(s).`);
}

return suggestions;
};
const formatCurrency = (amount) => {
if (!showValues) return '••••••';
return new Intl.NumberFormat('en-US', {
style: 'currency',
currency: 'USD'
}).format(amount);
};
const TabButton = ({ tab, label, icon: Icon }) => (
<button
onClick={() => setActiveTab(tab)}
className={`flex items-center space-x-2 px-4 py-2 rounded-lg transition-colors ${
activeTab === tab
? 'bg-blue-600 text-white'
: 'bg-gray-100 text-gray-700 hover:bg-gray-200'
}`}
>
<Icon size={18} />
<span>{label}</span>
</button>
);
return (
<div className="max-w-7xl mx-auto p-6 bg-gray-50 min-h-screen">
<div className="bg-white rounded-lg shadow-lg p-6 mb-6">
<div className="flex justify-between items-center mb-6">
<h1 className="text-3xl font-bold text-gray-800">Financial Dashboard</h1>
<button
onClick={() => setShowValues(!showValues)}
className="flex items-center space-x-2 px-4 py-2 bg-gray-200 rounded-lg hover:bg-gray-300 transition-colors"
>
{showValues ? <EyeOff size={18} /> : <Eye size={18} />}
<span>{showValues ? 'Hide' : 'Show'} Values</span>
</button>
</div>
{/* Overview Cards */}
<div className="grid grid-cols-1 md:grid-cols-4 gap-6 mb-8">
<div className="bg-green-50 p-6 rounded-lg border-l-4 border-green-500">
<div className="flex items-center justify-between">
<div>
<p className="text-green-600 text-sm font-medium">Net Worth</p>
<p className="text-2xl font-bold text-green-700">{formatCurrency(netWorth)}</p>
</div>
<TrendingUp className="text-green-500" size={24} />
</div>
```

```
</div>

<div className="bg-blue-50 p-6 rounded-lg border-l-4 border-blue-500">
<div className="flex items-center justify-between">
<div>
<p className="text-blue-600 text-sm font-medium">Total Assets</p>
<p className="text-2xl font-bold text-blue-700">{formatCurrency(totalAssets + portfolioValue)}</p>
</div>
<TrendingUp className="text-blue-500" size={24} />
</div>
</div>

<div className="bg-red-50 p-6 rounded-lg border-l-4 border-red-500">
<div className="flex items-center justify-between">
<div>
<p className="text-red-600 text-sm font-medium">Total Liabilities</p>
<p className="text-2xl font-bold text-red-700">{formatCurrency(totalLiabilities)}</p>
</div>
<TrendingDown className="text-red-500" size={24} />
</div>
</div>

<div className="bg-purple-50 p-6 rounded-lg border-l-4 border-purple-500">
<div className="flex items-center justify-between">
<div>
<p className="text-purple-600 text-sm font-medium">Portfolio Value</p>
<p className="text-2xl font-bold text-purple-700">{formatCurrency(portfolioValue)}</p>
</div>
<PieChart className="text-purple-500" size={24} />
</div>
</div>
</div>
{/* Navigation Tabs */}
<div className="flex flex-wrap gap-2 mb-6">
<TabButton tab="overview" label="Overview" icon={BarChart3} />
<TabButton tab="assets" label="Assets" icon={TrendingUp} />
<TabButton tab="liabilities" label="Liabilities" icon={TrendingDown} />
<TabButton tab="funds" label="Funds" icon={DollarSign} />
<TabButton tab="goals" label="Goals" icon={Target} />
<TabButton tab="portfolio" label="Portfolio" icon={PieChart} />
<TabButton tab="transactions" label="History" icon={BarChart3} />
<TabButton tab="suggestions" label="Suggestions" icon={Lightbulb} />
</div>
{/* Tab Content */}
{activeTab === 'overview' && (
<div className="space-y-6">
{/* Net Worth Visualization */}
<div className="grid grid-cols-1 lg:grid-cols-2 gap-6">
<div className="bg-white p-6 rounded-lg border">
<h3 className="text-lg font-semibold mb-4">Net Worth Breakdown</h3>
<div className="h-64">
<ResponsiveContainer width="100%" height="100%">
<RechartsPieChart>
<Pie
data={getNetWorthData()}
cx="50%"
cy="50%"
```

```
innerRadius={60}
outerRadius={100}
paddingAngle={5}
dataKey="value"
>
{getNetWorthData().map((entry, index) => (
<Cell key={`cell-${index}`} fill={entry.color} />
))}
</Pie>
<Tooltip content={<CustomTooltip />} />
<Legend />
</RechartsPieChart>
</ResponsiveContainer>
</div>
<div className="text-center mt-4">
<p className="text-sm text-gray-600">Net Worth</p>
<p className={`text-2xl font-bold ${netWorth >= 0 ? 'text-green-600' : 'text-red-600'}`}>
{formatCurrency(netWorth)}
</p>
</div>
</div>
{/* Assets by Category */}
<div className="bg-white p-6 rounded-lg border">
<h3 className="text-lg font-semibold mb-4">Assets by Category</h3>
<div className="h-64">
{getAssetsByCategory().length > 0 ? (
<ResponsiveContainer width="100%" height="100%">
<RechartsPieChart>
<Pie
data={getAssetsByCategory()}
cx="50%"
cy="50%"
outerRadius={100}
dataKey="value"
label={({ name, percent }) => `${name} ${(percent * 100).toFixed(0)}%`}
>
{getAssetsByCategory().map((entry, index) => (
<Cell key={`cell-${index}`} fill={COLORS[index % COLORS.length]} />
))}
</Pie>
<Tooltip content={<CustomTooltip />} />
</RechartsPieChart>
</ResponsiveContainer>
) : (
<div className="flex items-center justify-center h-full text-gray-500">
No assets added yet
</div>
)}
</div>
</div>
</div>
{/* Liabilities and Portfolio */}
<div className="grid grid-cols-1 lg:grid-cols-2 gap-6">
{/* Liabilities by Category */}
<div className="bg-white p-6 rounded-lg border">
<h3 className="text-lg font-semibold mb-4">Liabilities by Category</h3>
<div className="h-64">
```

```
{getLiabilitiesByCategory().length > 0 ? (
<ResponsiveContainer width="100%" height="100%">
<RechartsPieChart>
<Pie
data={getLiabilitiesByCategory()}
cx="50%"
cy="50%"
outerRadius={100}
dataKey="value"
label={({ name, percent }) => ${name} ${(percent * 100).toFixed(0)}%}
>
{getLiabilitiesByCategory().map((entry, index) => (
<Cell key={`cell-${index}`} fill={COLORS[index % COLORS.length]} />
))}
</Pie>
<Tooltip content={<CustomTooltip />} />
</RechartsPieChart>
</ResponsiveContainer>
) : (
<div className="flex items-center justify-center h-full text-gray-500">
No liabilities added yet
</div>
)}
</div>
</div>
{/* Portfolio Performance */}
<div className="bg-white p-6 rounded-lg border">
<h3 className="text-lg font-semibold mb-4">Portfolio Performance</h3>
<div className="h-64">
{getPortfolioData().length > 0 ? (
<ResponsiveContainer width="100%" height="100%">
<BarChart data={getPortfolioData()}>
<CartesianGrid strokeDasharray="3 3" />
<XAxis dataKey="symbol" />
<YAxis />
<Tooltip
formatter={(value, name) => [
name === 'gainLoss' ? formatCurrency(value) : value,
name === 'gainLoss' ? 'Gain/Loss' : 'Value'
]}
/>
<Bar dataKey="value" fill="#3B82F6" name="Current Value" />
<Bar dataKey="gainLoss" fill="#10B981" name="Gain/Loss" />
</BarChart>
</ResponsiveContainer>
) : (
<div className="flex items-center justify-center h-full text-gray-500">
No portfolio positions yet
</div>
)}
</div>
</div>
</div>
{/* Summary Tables */}
<div className="grid grid-cols-1 lg:grid-cols-2 gap-6">
<div className="bg-gray-50 p-6 rounded-lg">
<h3 className="text-lg font-semibold mb-4">Asset Summary</h3>
```

```
{getAssetsByCategory().length === 0 ? (
<p className="text-gray-500">No assets added yet</p>
) : (
<div className="space-y-2">
{getAssetsByCategory().map((category, index) => (
<div key={category.name} className="flex justify-between items-center">
<div className="flex items-center">
<div
className="w-4 h-4 rounded mr-2"
style={{ backgroundColor: COLORS[index % COLORS.length] }}
></div>
<span className="text-sm">{ category.name}</span>
</div>
<span className="font-medium">{formatCurrency(category.value)}</span>
</div>
))}
<div className="border-t pt-2 mt-2">
<div className="flex justify-between items-center font-semibold">
<span>Total Assets</span>
<span>{formatCurrency(totalAssets + portfolioValue)}</span>
</div>
</div>
</div>
)}
</div>

<div className="bg-gray-50 p-6 rounded-lg">
<h3 className="text-lg font-semibold mb-4">Recent Activity</h3>
{transactions.slice(0, 5).map(transaction => (
<div key={transaction.id} className="flex justify-between items-center mb-2 text-sm">
<span>{transaction.action} {transaction.item}</span>
<span className="text-gray-500">{new Date( transaction.date).toLocaleDateString()}</span>
</div>
))}
{transactions.length === 0 && (
<p className="text-gray-500">No recent activity</p>
)}
</div>
</div>
</div>
)}
{activeTab === 'assets' && (
<div>
<div className="mb-6 p-4 bg-gray-50 rounded-lg">
<h3 className="text-lg font-semibold mb-4">Add New Asset</h3>
<div className="grid grid-cols-1 md:grid-cols-4 gap-4">
<input
type="text"
placeholder="Asset name"
value={assetForm.name}
onChange={(e) => setAssetForm({...assetForm, name: e.target.value})}
className="border rounded-lg px-3 py-2"
/>
<input
type="number"
placeholder="Value"
value={assetForm.value}
```

```
onChange={(e) => setAssetForm({...assetForm, value: e.target.value})}
className="border rounded-lg px-3 py-2"
/>
<select
value={assetForm.type}
onChange={(e) => setAssetForm({...assetForm, type: e.target.value})}
className="border rounded-lg px-3 py-2"
>
<option value="Real Estate">Real Estate</option>
<option value="Cash">Cash</option>
<option value="Investment">Investment</option>
<option value="Vehicle">Vehicle</option>
<option value="Other">Other</option>
</select>
<button
onClick={addAsset}
className="bg-blue-600 text-white px-4 py-2 rounded-lg hover:bg-blue-700 flex items-center justify-center"
>
<PlusCircle size={18} className="mr-2" />
Add Asset
</button>
</div>
</div>
<div className="space-y-4">
{ assets.map(asset => (
<div key={asset.id} className="flex justify-between items-center p-4 bg-white border rounded-lg">
<div>
<h4 className="font-medium">{ asset.name}</h4>
<p className="text-sm text-gray-600">{asset.type}</p>
</div>
<div className="flex items-center space-x-4">
<span className="font-medium text-lg">{formatCurrency(asset.value)}</span>
<button
onClick={() => removeAsset( asset.id)}
className="text-red-600 hover:text-red-800"
>
<Trash2 size={18} />
</button>
</div>
</div>
))}
</div>
</div>
)}
{activeTab === 'liabilities' && (
<div>
<div className="mb-6 p-4 bg-gray-50 rounded-lg">
<h3 className="text-lg font-semibold mb-4">Add New Liability</h3>
<div className="grid grid-cols-1 md:grid-cols-4 gap-4">
<input
type="text"
placeholder="Liability name"
value={liabilityForm.name}
onChange={(e) => setLiabilityForm({...liabilityForm, name: e.target.value})}
className="border rounded-lg px-3 py-2"
/>
```

```
<input
type="number"
placeholder="Amount"
value={liabilityForm.amount}
onChange={(e) => setLiabilityForm({...liabilityForm, amount: e.target.value})}
className="border rounded-lg px-3 py-2"
/>
<select
value={liabilityForm.type}
onChange={(e) => setLiabilityForm({...liabilityForm, type: e.target.value})}
className="border rounded-lg px-3 py-2"
>
<option value="Loan">Loan</option>
<option value="Credit Card">Credit Card</option>
<option value="Mortgage">Mortgage</option>
<option value="Other">Other</option>
</select>
<button
onClick={addLiability}
className="bg-red-600 text-white px-4 py-2 rounded-lg hover:bg-red-700 flex items-center justify-
center"
>
<PlusCircle size={18} className="mr-2" />
Add Liability
</button>
</div>
</div>
<div className="space-y-4">
{ liabilities.map(liability => (
<div key={liability.id} className="flex justify-between items-center p-4 bg-white border rounded-lg">
<div>
<h4 className="font-medium">{ liability.name}</h4>
<p className="text-sm text-gray-600">{liability.type}</p>
</div>
<div className="flex items-center space-x-4">
<span className="font-medium text-lg text-red-600">{formatCurrency(liability.amount)}</span>
<button
onClick={() => removeLiability( liability.id)}
className="text-red-600 hover:text-red-800"
>
<Trash2 size={18} />
</button>
</div>
</div>
))}
</div>
</div>
)}
{activeTab === 'funds' && (
<div>
<div className="mb-6 p-4 bg-gray-50 rounded-lg">
<h3 className="text-lg font-semibold mb-4">Create New Fund</h3>
<div className="grid grid-cols-1 md:grid-cols-4 gap-4">
<input
type="text"
placeholder="Fund name"
value={fundForm.name}
```

```
onChange={(e) => setFundForm({...fundForm, name: e.target.value})}
className="border rounded-lg px-3 py-2"
/>
<input
type="number"
placeholder="Initial balance"
value={fundForm.balance}
onChange={(e) => setFundForm({...fundForm, balance: e.target.value})}
className="border rounded-lg px-3 py-2"
/>
<input
type="number"
placeholder="Target (optional)"
value={fundForm.target}
onChange={(e) => setFundForm({...fundForm, target: e.target.value})}
className="border rounded-lg px-3 py-2"
/>
<button
onClick={addFund}
className="bg-green-600 text-white px-4 py-2 rounded-lg hover:bg-green-700 flex items-center jus-
tify-center"
>
<PlusCircle size={18} className="mr-2" />
Create Fund
</button>
</div>
</div>
<div className="space-y-4">
{ funds.map(fund => (
<div key={fund.id} className="p-4 bg-white border rounded-lg">
<div className="flex justify-between items-center mb-2">
<h4 className="font-medium">{ fund.name}</h4>
<button
onClick={() => removeFund( fund.id)}
className="text-red-600 hover:text-red-800"
>
<Trash2 size={18} />
</button>
</div>
<div className="flex justify-between items-center mb-3">
<span className="text-lg font-medium">{formatCurrency(fund.balance)}</span>
{ fund.target > 0 && (
<span className="text-sm text-gray-600">
Target: {formatCurrency( fund.target)} ({Math.round((fund.balance / fund.target) * 100)}%)
</span>
)}
</div>
{ fund.target > 0 && (
<div className="w-full bg-gray-200 rounded-full h-2 mb-3">
<div
className="bg-green-600 h-2 rounded-full"
style={{ width: ${Math.min((fund.balance / fund.target) * 100, 100)}% }}
></div>
</div>
)}
<div className="flex gap-2">
<input
```

```
type="number"
placeholder="Contribution amount"
className="border rounded-lg px-3 py-1 flex-1"
onKeyPress={(e) => {
if (e.key === 'Enter') {
contributeFund( fund.id, e.target.value);
e.target.value = '';
}
}}
/>
<button
onClick={(e) => {
const input = e.target.previousSibling;
contributeFund( fund.id, input.value);
input.value = '';
}}
className="bg-blue-600 text-white px-4 py-1 rounded-lg hover:bg-blue-700"
>
Contribute
</button>
</div>
</div>
))}
</div>
</div>
)}
{activeTab === 'goals' && (
<div>
<div className="mb-6 p-4 bg-gray-50 rounded-lg">
<h3 className="text-lg font-semibold mb-4">Set New Goal</h3>
<div className="grid grid-cols-1 md:grid-cols-4 gap-4">
<input
type="text"
placeholder="Goal name"
value={goalForm.name}
onChange={(e) => setGoalForm({...goalForm, name: e.target.value})}
className="border rounded-lg px-3 py-2"
/>
<input
type="number"
placeholder="Target amount"
value={goalForm.target}
onChange={(e) => setGoalForm({...goalForm, target: e.target.value})}
className="border rounded-lg px-3 py-2"
/>
<input
type="number"
placeholder="Current progress"
value={goalForm.current}
onChange={(e) => setGoalForm({...goalForm, current: e.target.value})}
className="border rounded-lg px-3 py-2"
/>
<button
onClick={addGoal}
className="bg-purple-600 text-white px-4 py-2 rounded-lg hover:bg-purple-700 flex items-center
justify-center"
>
```

```
<Target size={18} className="mr-2" />
Set Goal
</button>
</div>
</div>
<div className="space-y-4">
{ goals.map(goal => (
<div key={goal.id} className="p-4 bg-white border rounded-lg">
<div className="flex justify-between items-center mb-2">
<h4 className="font-medium">{ goal.name}</h4>
<button
onClick={() => removeGoal( goal.id)}
className="text-red-600 hover:text-red-800"
>
<Trash2 size={18} />
</button>
</div>
<div className="flex justify-between items-center mb-3">
<span className="text-lg font-medium">{formatCurrency(goal.current)}</span>
<span className="text-sm text-gray-600">
Target: {formatCurrency( goal.target)} ({Math.round((goal.current / goal.target) * 100)}%)
</span>
</div>
<div className="w-full bg-gray-200 rounded-full h-2 mb-3">
<div
className={`h-2 rounded-full ${
goal.current >= goal.target ? 'bg-green-600' : 'bg-blue-600'
}`}
style={{ width: ${Math.min((goal.current / goal.target) * 100, 100)}% }}
></div>
</div>
{goal.current >= goal.target && (
<div className="text-green-600 text-sm font-medium mb-2">🎉 Goal Achieved!</div>
)}
<div className="flex gap-2">
<input
type="number"
placeholder="Progress update"
className="border rounded-lg px-3 py-1 flex-1"
onKeyPress={(e) => {
if (e.key === 'Enter') {
updateGoal( goal.id, e.target.value);
e.target.value = '';
}
}}
/>
<button
onClick={(e) => {
const input = e.target.previousSibling;
updateGoal( goal.id, input.value);
input.value = '';
}}
className="bg-blue-600 text-white px-4 py-1 rounded-lg hover:bg-blue-700"
>
Update
</button>
</div>
```

```
</div>
))}
</div>
</div>
)}
{activeTab === 'portfolio' && (
<div>
<div className="mb-6 p-4 bg-gray-50 rounded-lg">
<h3 className="text-lg font-semibold mb-4">Add Stock Position</h3>
<div className="grid grid-cols-1 md:grid-cols-5 gap-4">
<input
type="text"
placeholder="Stock symbol"
value={portfolioForm.symbol}
onChange={(e) => setPortfolioForm({...portfolioForm, symbol: e.target.value.toUpperCase()})}
className="border rounded-lg px-3 py-2"
/>
<input
type="number"
placeholder="Shares"
value={portfolioForm.shares}
onChange={(e) => setPortfolioForm({...portfolioForm, shares: e.target.value})}
className="border rounded-lg px-3 py-2"
/>
<input
type="number"
step="0.01"
placeholder="Avg price"
value={portfolioForm.avgPrice}
onChange={(e) => setPortfolioForm({...portfolioForm, avgPrice: e.target.value})}
className="border rounded-lg px-3 py-2"
/>
<input
type="number"
step="0.01"
placeholder="Current price"
value={portfolioForm.currentPrice}
onChange={(e) => setPortfolioForm({...portfolioForm, currentPrice: e.target.value})}
className="border rounded-lg px-3 py-2"
/>
<button
onClick={addStock}
className="bg-purple-600 text-white px-4 py-2 rounded-lg hover:bg-purple-700 flex items-center
justify-center"
>
<PlusCircle size={18} className="mr-2" />
Add Stock
</button>
</div>
</div>
{/* Portfolio Visual Overview */}
{portfolio.length > 0 && (
<div className="mb-6 bg-white p-6 rounded-lg border">
<h3 className="text-lg font-semibold mb-4">Portfolio Allocation</h3>
<div className="h-80">
<ResponsiveContainer width="100%" height="100%">
<RechartsPieChart>
```

```
<Pie
data={getPortfolioData()}
cx="50%"
cy="50%"
outerRadius={120}
dataKey="value"
label={({ symbol, value }) => `${symbol}: ${formatCurrency(value)}`}
>
{getPortfolioData().map((entry, index) => (
<Cell key={`cell-${index}`} fill={COLORS[index % COLORS.length]} />
))}
</Pie>
<Tooltip
formatter={(value, name, props) => [
formatCurrency(value),
'Value'
]}
/>
</RechartsPieChart>
</ResponsiveContainer>
</div>
</div>
)}
<div className="space-y-4">
{ portfolio.map(stock => {
const totalValue = stock.shares * stock.currentPrice;
const totalCost = stock.shares * stock.avgPrice;
const gainLoss = totalValue - totalCost;
const gainLossPercent = ((stock.currentPrice - stock.avgPrice) / stock.avgPrice) * 100;

return (
<div key={stock.id} className="p-4 bg-white border rounded-lg">
<div className="flex justify-between items-center mb-2">
<div>
<h4 className="font-medium text-lg">{stock.symbol}</h4>
<p className="text-sm text-gray-600">{stock.shares} shares @ {formatCurrency(stock.avgPrice)}
avg</p>
</div>
<button
onClick={() => removeStock( stock.id)}
className="text-red-600 hover:text-red-800"
>
<Trash2 size={18} />
</button>
</div>

<div className="grid grid-cols-2 md:grid-cols-4 gap-4 text-sm">
<div>
<p className="text-gray-600">Current Price</p>
<p className="font-medium">{formatCurrency(stock.currentPrice)}</p>
</div>
<div>
<p className="text-gray-600">Market Value</p>
<p className="font-medium">{formatCurrency(totalValue)}</p>
</div>
<div>
<p className="text-gray-600">Total Cost</p>
```

```
<p className="font-medium">{formatCurrency(totalCost)}</p>
</div>
<div>
<p className="text-gray-600">Gain/Loss</p>
<p className={`font-medium ${gainLoss >= 0 ? 'text-green-600' : 'text-red-600'}`}>
{formatCurrency(gainLoss)} ({gainLossPercent >= 0 ? '+' : ''}{gainLossPercent.toFixed(2)}%)
</p>
</div>
</div>

<div className="mt-3 flex gap-2">
<input
type="number"
step="0.01"
placeholder="Update current price"
className="border rounded-lg px-3 py-1 flex-1"
onKeyPress={(e) => {
if (e.key === 'Enter') {
const newPrice = parseFloat( e.target.value);
if (newPrice > 0) {
setPortfolio(prev =>
prev.map(s =>
s.id === stock.id
? { ...s, currentPrice: newPrice }
: s
)
);
addTransaction('Portfolio', 'Price Updated', stock, newPrice);
e.target.value = '';
}
}
}}
/>
<button
onClick={(e) => {
const input = e.target.previousSibling;
const newPrice = parseFloat(input.value);
if (newPrice > 0) {
setPortfolio(prev =>
prev.map(s =>
s.id === stock.id
? { ...s, currentPrice: newPrice }
: s
)
);
addTransaction('Portfolio', 'Price Updated', stock, newPrice);
input.value = '';
}
}}
className="bg-blue-600 text-white px-4 py-1 rounded-lg hover:bg-blue-700"
>
Update Price
</button>
</div>
</div>
);
})}
```

```
</div>
</div>
)}
{activeTab === 'transactions' && (
<div>
<h3 className="text-lg font-semibold mb-4">Transaction History</h3>
<div className="space-y-2">
{ transactions.map(transaction => (
<div key={transaction.id} className="flex justify-between items-center p-3 bg-white border
rounded-lg">
<div>
<span className="font-medium">{transaction.action}</span>
<span className="text-gray-600 ml-2">{transaction.item}</span>
<div className="text-sm text-gray-500">
{transaction.type} • {new Date( transaction.date).toLocaleString()}
</div>
</div>
<div className="text-right">
<span className="font-medium">{formatCurrency(transaction.amount)}</span>
</div>
</div>
))}
{transactions.length === 0 && (
<p className="text-gray-500 text-center py-4">No transactions yet</p>
)}
</div>
</div>
)}
{activeTab === 'suggestions' && (
<div>
<h3 className="text-lg font-semibold mb-4">Financial Suggestions</h3>
<div className="space-y-4">
{getFinancialSuggestions().map((suggestion, index) => (
<div key={index} className="p-4 bg-yellow-50 border-l-4 border-yellow-400 rounded-lg">
<div className="flex items-start">
<Lightbulb className="text-yellow-600 mr-3 mt-1" size={20} />
<p className="text-yellow-800">{suggestion}</p>
</div>
</div>
))}
{getFinancialSuggestions().length === 0 && (
<div className="p-4 bg-green-50 border-l-4 border-green-400 rounded-lg">
<div className="flex items-start">
<Lightbulb className="text-green-600 mr-3 mt-1" size={20} />
<p className="text-green-800">Great job! Your financial profile looks healthy. Keep monitoring your
progress regularly.</p>
</div>
</div>
)}
</div>
</div>
)}
</div>
</div>
);
};
export default FinancialTracker
```