Investment Portfolio Frontend Code

This document contains all the code files for the Investment Portfolio application for printing purposes.

Table of Contents

Project Configuration

package.json

```
{
  "name": "investment-portfolio",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^1.3.4",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-router-dom": "^6.8.2",
    "react-scripts": "5.0.1",
    "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"
    ]
  }
}
```

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="Investment Portfolio App"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <title>Investment Portfolio</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

Entry Points

src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

reportWebVitals();
```

src/index.css

```
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}
```

src/reportWebVitals.js

```
const reportWebVitals = (onPerfEntry) => {
  if (onPerfEntry && onPerfEntry instanceof Function) {
    import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB }) => {
      getCLS(onPerfEntry);
      getFID(onPerfEntry);
      getFCP(onPerfEntry);
      getLCP(onPerfEntry);
      getTTFB(onPerfEntry);
    });
  }
};

export default reportWebVitals;
```

src/App.js

```
import React, { useState } from 'react';
import { BrowserRouter as Router, Routes, Route, Navigate } from 'react-router-dom';
import Navbar from './components/Navbar';
import Dashboard from './components/Dashboard';
import CompanyList from './components/CompanyList';
import CompanyDetail from './components/CompanyDetail';
import Login from './components/Login';
import Register from './components/Register';
import './App.css';

function App() {
  const [user, setUser] = useState(null);

  return (
    <Router>
      <div className="App">
        <Navbar user={user} setUser={setUser} />
        <main className="main-content">
          <Routes>
            <Route path="/login" element={
              !user ? <Login setUser={setUser} /> : <Navigate to="/dashboard" />
            } />
            <Route path="/register" element={
              !user ? <Register /> : <Navigate to="/dashboard" />
            } />
            <Route path="/dashboard" element={
              user ? <Dashboard user={user} /> : <Navigate to="/login" />
            } />
            <Route path="/companies" element={<CompanyList />} />
            <Route path="/company/:symbol" element={<CompanyDetail />} />
            <Route path="/" element={<Navigate to="/companies" />} />
          </Routes>
        </main>
      </div>
    </Router>
  );
}

export default App;
```

src/App.css

```
/* Global Styles */
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}

body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  line-height: 1.6;
  color: #333;
  background-color: #f5f7fa;
}
```

```css
.App {
  display: flex;
  flex-direction: column;
  min-height: 100vh;
}

.main-content {
  padding: 20px;
  max-width: 1200px;
  margin: 0 auto;
  width: 100%;
  flex: 1;
}

button {
  cursor: pointer;
  background-color: #4a6fa5;
  color: white;
  border: none;
  padding: 8px 15px;
  border-radius: 4px;
  font-size: 0.9rem;
  transition: background-color 0.3s;
}

button:hover {
  background-color: #385987;
}

input, select {
  padding: 8px 12px;
  border: 1px solid #ccd;
  border-radius: 4px;
  font-size: 0.9rem;
}

/* Navbar Styles */
.navbar {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 0 20px;
  background-color: #1f2942;
  color: white;
  height: 60px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.navbar-brand a {
  color: white;
  font-size: 1.4rem;
  font-weight: bold;
  text-decoration: none;
  display: flex;
  align-items: center;
}

.logo-icon {
  margin-right: 8px;
}

.navbar-menu {
  display: flex;
  align-items: center;
  gap: 20px;
}

.navbar-menu a {
  color: #e0e0e0;
  text-decoration: none;
  font-size: 0.95rem;
  display: flex;
  align-items: center;
  padding: 5px 8px;
  border-radius: 4px;
  transition: all 0.3s;
}

.navbar-menu a:hover, .navbar-menu a.active {
  color: white;
  background-color: rgba(255, 255, 255, 0.1);
}

.nav-icon {
  margin-right: 6px;
}

.logout-btn {
  background-color: transparent;
  padding: 5px 8px;
  font-size: 0.95rem;
}

.logout-btn:hover {
  background-color: rgba(255, 255, 255, 0.1);
}

.user-info {
  display: flex;
  align-items: center;
  margin-left: 10px;
}
```

```css
.user-avatar {
  width: 30px;
  height: 30px;
  border-radius: 50%;
  background-color: #4a6fa5;
  display: flex;
  align-items: center;
  justify-content: center;
  margin-right: 8px;
  font-size: 0.8rem;
  font-weight: bold;
}

/* Dashboard Styles */
.dashboard {
  display: flex;
  flex-direction: column;
  gap: 20px;
}

.dashboard-section {
  background-color: white;
  border-radius: 8px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.05);
  padding: 20px;
  margin-bottom: 20px;
}

.section-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 15px;
}

.last-updated {
  color: #888;
  font-size: 0.8rem;
}

/* Company List Styles */
.companies-container {
  display: flex;
  flex-direction: column;
  gap: 20px;
}

.page-header {
  margin-bottom: 20px;
}

.filter-controls {
  display: flex;
  gap: 15px;
  margin-bottom: 20px;
}

.filter-container {
  position: relative;
  flex: 1;
}

.search-icon {
  position: absolute;
  left: 10px;
  top: 50%;
  transform: translateY(-50%);
  color: #888;
}

.search-input {
  width: 100%;
  padding-left: 30px;
}

.companies-grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(280px, 1fr));
  gap: 20px;
}

.company-card {
  background-color: white;
  border-radius: 8px;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.05);
  padding: 15px;
  text-decoration: none;
  color: inherit;
  transition: transform 0.2s, box-shadow 0.2s;
}

.company-card:hover {
  transform: translateY(-3px);
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
}

.company-card-header {
  display: flex;
  justify-content: space-between;
  align-items: flex-start;
  margin-bottom: 10px;
}

.company-card h3 {
```

```
  margin: 0;
  font-size: 1.1rem;
}

.symbol {
  background-color: #f0f4f9;
  padding: 2px 6px;
  border-radius: 4px;
  font-size: 0.8rem;
  font-weight: bold;
  color: #4a6fa5;
}

/* Company Detail Styles */
.company-detail {
  display: flex;
  flex-direction: column;
  gap: 20px;
}

.company-header {
  display: flex;
  flex-direction: column;
  gap: 5px;
  margin-bottom: 10px;
}

.detail-grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(350px, 1fr));
  gap: 20px;
}

.detail-card {
  background-color: white;
  border-radius: 8px;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.05);
  padding: 20px;
}

.data-grid {
  display: grid;
  grid-template-columns: repeat(2, 1fr);
  gap: 15px;
  margin-top: 15px;
}

.data-item label {
  display: block;
  font-size: 0.8rem;
  color: #777;
  margin-bottom: 4px;
}

/* Login & Register Styles */
.login-container, .register-container {
  max-width: 400px;
  margin: 40px auto;
  background-color: white;
  padding: 30px;
  border-radius: 8px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
}

.form-group {
  margin-bottom: 20px;
}

.form-group label {
  display: block;
  margin-bottom: 8px;
  font-weight: 500;
}

.form-group input {
  width: 100%;
}

.login-button, .register-button {
  width: 100%;
  padding: 10px;
  font-size: 1rem;
  margin-top: 10px;
}

.register-link, .login-link {
  text-align: center;
  margin-top: 20px;
  font-size: 0.9rem;
}

.register-link a, .login-link a {
  color: #4a6fa5;
  text-decoration: none;
}

.error-message {
  background-color: #fee;
  color: #c00;
  padding: 10px;
  border-radius: 4px;
  margin-bottom: 15px;
  font-size: 0.9rem;
}
```

API Service

src/services/api.js

```
import axios from 'axios';

// Base URL for API requests
const API_BASE_URL = 'http://localhost:5000/api';

// Create axios instance
const apiClient = axios.create({
  baseURL: API_BASE_URL,
  headers: {
    'Content-Type': 'application/json'
  }
});

// Add auth token to requests
apiClient.interceptors.request.use(config => {
  const token = localStorage.getItem('token');
  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }
  return config;
}, error => {
  return Promise.reject(error);
});

// API methods
const api = {
  // Authentication
  login: async (credentials) => {
    return await apiClient.post('/auth/login', credentials);
  },

  register: async (userData) => {
    return await apiClient.post('/auth/register', userData);
  },

  // Companies
  getCompanies: async () => {
    return await apiClient.get('/companies');
  },

  getCompanyDetails: async (symbol) => {
    return await apiClient.get(`/companies/${symbol}`);
  },

  // Portfolio
  getPortfolio: async () => {
    return await apiClient.get('/portfolio');
  },

  // Transactions
  getTransactions: async () => {
    return await apiClient.get('/transactions');
  },

  addTransaction: async (transaction) => {
    return await apiClient.post('/transactions', transaction);
  },

  // Watchlist
  getWatchlist: async () => {
    return await apiClient.get('/watchlist');
  },

  addToWatchlist: async (symbol) => {
    return await apiClient.post('/watchlist', { symbol });
  },

  removeFromWatchlist: async (symbol) => {
    return await apiClient.delete(`/watchlist/${symbol}`);
  }
};

export default api;
```

Components

src/components/Navbar.js

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

function Navbar({ user, setUser }) {
  const navigate = useNavigate();
  const location = useLocation();

  const handleLogout = () => {
    setUser(null);
    localStorage.removeItem('token');
    navigate('/login');
  };

  const getInitials = (name) => {
    if (!name) return 'U';
    return name.split(' ').map(n => n[0]).join('').toUpperCase();
  };

  const isActive = (path) => {
    return location.pathname === path ? 'active' : '';
  };

  return (
    <nav className="navbar">
      <div className="navbar-brand">
        <Link to="/">
          <span className="logo-icon">📈</span> Investment Portfolio
        </Link>
      </div>
      <div className="navbar-menu">
        <Link to="/companies" className={isActive('/companies')}>
          <span className="nav-icon">🏢</span> Companies
        </Link>

        {user ? (
          <>
            <Link to="/dashboard" className={isActive('/dashboard')}>
              <span className="nav-icon">📊</span> Dashboard
            </Link>
            <button onClick={handleLogout} className="logout-btn">
              <span className="nav-icon">🚪</span> Logout
            </button>
            <div className="user-info">
              <div className="user-avatar">
                {getInitials(user.name || user.username)}
              </div>
              <span className="username">{user.name || user.username}</span>
            </div>
          </>
        ) : (
          <Link to="/login" className={isActive('/login')}>
            <span className="nav-icon">🔑</span> Login
          </Link>
        )}
      </div>
    </nav>
  );
}

export default Navbar;
```

src/components/Login.js

```
import React, { useState } from 'react';
import { useNavigate, Link } from 'react-router-dom';
import api from '../services/api';

function Login({ setUser }) {
  const [credentials, setCredentials] = useState({ username: '', password: '' });
  const [error, setError] = useState('');
  const [loading, setLoading] = useState(false);
  const navigate = useNavigate();

  const handleSubmit = async (e) => {
    e.preventDefault();
    setLoading(true);
    setError('');

    try {
      const response = await api.login(credentials);

      // Check if we have a valid user object
      if (!response.data || !response.data.user) {
        setError('Invalid user data received from server');
        setLoading(false);
        return;
      }

      // Save token to local storage
      if (response.data.token) {
        localStorage.setItem('token', response.data.token);
      }

      // Set user data and navigate to dashboard
      setUser(response.data.user);
      navigate('/dashboard');
    } catch (err) {
      setError(
        err.response?.data?.error ||
        err.message ||
        'Login failed. Please try again.'
      );
      setLoading(false);
    }
  };

  const handleChange = (e) => {
    setCredentials({
      ...credentials,
      [e.target.name]: e.target.value
    });
  };

  return (
    <div className="login-container">
      <h2>Login</h2>
      {error && <div className="error-message">{error}</div>}
      <form onSubmit={handleSubmit} className="login-form">
        <div className="form-group">
          <label htmlFor="username">Username:</label>
          <input
            type="text"
            id="username"
            name="username"
            value={credentials.username}
            onChange={handleChange}
            disabled={loading}
            required
          />
        </div>
        <div className="form-group">
          <label htmlFor="password">Password:</label>
          <input
            type="password"
            id="password"
            name="password"
            value={credentials.password}
            onChange={handleChange}
            disabled={loading}
            required
          />
        </div>
        <button
          type="submit"
          className="login-button"
          disabled={loading}
        >
          {loading ? 'Logging in...' : 'Login'}
        </button>
      </form>
      <p className="register-link">
        Don't have an account? <Link to="/register">Register here</Link>
      </p>
    </div>
  );
}

export default Login;
```

src/components/Register.js

```
import React, { useState } from 'react';
import { useNavigate, Link } from 'react-router-dom';
import api from '../services/api';

function Register() {
```

```
const [userData, setUserData] = useState({
  username: '',
  email: '',
  password: '',
  confirmPassword: ''
});
const [errors, setErrors] = useState({});
const [loading, setLoading] = useState(false);
const [serverError, setServerError] = useState('');
const navigate = useNavigate();

const validate = () => {
  const newErrors = {};

  if (!userData.username || userData.username.length < 3) {
    newErrors.username = 'Username must be at least 3 characters';
  }

  if (!userData.email || !/\S+@\S+\.\S+/.test(userData.email)) {
    newErrors.email = 'Valid email is required';
  }

  if (!userData.password || userData.password.length < 6) {
    newErrors.password = 'Password must be at least 6 characters';
  }

  if (userData.password !== userData.confirmPassword) {
    newErrors.confirmPassword = 'Passwords do not match';
  }

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

const handleSubmit = async (e) => {
  e.preventDefault();

  if (!validate()) {
    return;
  }

  setLoading(true);
  setServerError('');

  try {
    const { confirmPassword, ...registerData } = userData;
    await api.register(registerData);
    navigate('/login');
  } catch (err) {
    setServerError(
      err.response?.data?.error ||
      err.message ||
      'Registration failed. Please try again.'
    );
    setLoading(false);
  }
};

const handleChange = (e) => {
  setUserData({
    ...userData,
    [e.target.name]: e.target.value
  });

  // Clear error when user starts typing
  if (errors[e.target.name]) {
    const { [e.target.name]: _, ...rest } = errors;
    setErrors(rest);
  }
};

return (
  <div className="register-container">
    <h2>Create Account</h2>
    {serverError && <div className="error-message">{serverError}</div>}

    <form onSubmit={handleSubmit} className="register-form">
      <div className="form-group">
        <label htmlFor="username">Username:</label>
        <input
          type="text"
          id="username"
          name="username"
          value={userData.username}
          onChange={handleChange}
          disabled={loading}
          required
        />
        {errors.username && <div className="field-error">{errors.username}</div>}
      </div>

      <div className="form-group">
        <label htmlFor="email">Email:</label>
        <input
          type="email"
          id="email"
          name="email"
          value={userData.email}
          onChange={handleChange}
          disabled={loading}
          required
        />
        {errors.email && <div className="field-error">{errors.email}</div>}
      </div>
```

```
        <div className="form-group">
          <label htmlFor="password">Password:</label>
          <input
            type="password"
            id="password"
            name="password"
            value={userData.password}
            onChange={handleChange}
            disabled={loading}
            required
          />
          {errors.password && <div className="field-error">{errors.password}</div>}
        </div>

        <div className="form-group">
          <label htmlFor="confirmPassword">Confirm Password:</label>
          <input
            type="password"
            id="confirmPassword"
            name="confirmPassword"
            value={userData.confirmPassword}
            onChange={handleChange}
            disabled={loading}
            required
          />
          {errors.confirmPassword && <div className="field-error">{errors.confirmPassword}</div>}
        </div>

        <button
          type="submit"
          className="register-button"
          disabled={loading}
        >
          {loading ? 'Creating Account...' : 'Register'}
        </button>
      </form>

      <p className="login-link">
        Already have an account? <Link to="/login">Login here</Link>
      </p>
    </div>
  );
}

export default Register;
```

src/components/CompanyList.js

```
import React, { useState, useEffect } from 'react';
import { Link } from 'react-router-dom';
import api from '../services/api';

function CompanyList() {
  const [companies, setCompanies] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [filter, setFilter] = useState('');
  const [sectors, setSectors] = useState([]);
  const [selectedSector, setSelectedSector] = useState('');

  useEffect(() => {
    const fetchCompanies = async () => {
      try {
        const response = await api.getCompanies();
        setCompanies(response.data);

        // Extract unique sectors
        const uniqueSectors = [...new Set(response.data.map(company => company.Sector).filter(Boolean))];
        setSectors(uniqueSectors);

        setLoading(false);
      } catch (err) {
        setError(err.message);
        setLoading(false);
      }
    };

    fetchCompanies();
  }, []);

  const filteredCompanies = companies.filter(company => {
    const matchesSearch =
      company.Company_name?.toLowerCase().includes(filter.toLowerCase()) ||
      company.Symbol?.toLowerCase().includes(filter.toLowerCase()) ||
      company.Sector?.toLowerCase().includes(filter.toLowerCase());

    const matchesSector = selectedSector === '' || company.Sector === selectedSector;

    return matchesSearch && matchesSector;
  });

  if (loading) return (
    <div className="loading">
      <div className="loading-spinner"></div>
      <p>Loading companies...</p>
    </div>
  );

  if (error) return <div className="error">Error: {error}</div>;

  return (
    <div className="companies-container">
      <div className="page-header">
```

```
        <h1>Companies</h1>
        <p>Browse and search companies to view details and add to your portfolio</p>
      </div>

      <div className="filter-controls">
        <div className="filter-container">
          <span className="search-icon">🔍</span>
          <input
            type="text"
            placeholder="Search companies by name, symbol or sector..."
            value={filter}
            onChange={(e) => setFilter(e.target.value)}
            className="search-input"
          />
        </div>

        <div className="sector-filter">
          <label htmlFor="sector-select">Filter by Sector:</label>
          <select
            id="sector-select"
            value={selectedSector}
            onChange={(e) => setSelectedSector(e.target.value)}
          >
            <option value="">All Sectors</option>
            {sectors.map(sector => (
              <option key={sector} value={sector}>{sector}</option>
            ))}
          </select>
        </div>
      </div>

      {selectedSector && (
        <div className="active-filters">
          <span className="filter-tag">
            {selectedSector}
            <button onClick={() => setSelectedSector('')}>×</button>
          </span>
        </div>
      )}

      <div className="companies-count">
        <span>{filteredCompanies.length} companies found</span>
      </div>

      <div className="companies-grid">
        {filteredCompanies.map((company) => (
          <Link to={`/company/${company.Symbol}`} key={company.Symbol} className="company-card">
            <div className="company-card-header">
              <h3>{company.Company_name}</h3>
              <span className="symbol">{company.Symbol}</span>
            </div>
            {company.Sector && <p className="sector">📊 {company.Sector}</p>}
            {company.Market_cap && (
              <p className="market-cap">
                <span className="label">Market Cap:</span> ${company.Market_cap.toLocaleString()}
              </p>
            )}
            {company.Listed_share && (
              <p className="listed-shares">
                <span className="label">Listed Shares:</span> {company.Listed_share.toLocaleString()}
              </p>
            )}
          </Link>
        ))}
      </div>

      {filteredCompanies.length === 0 && (
        <div className="no-results">
          <h3>No companies found matching your search</h3>
          <p>Try adjusting your search criteria or select a different sector</p>
          <button onClick={() => {setFilter(''); setSelectedSector('')}} className="reset-button">
            Reset Filters
          </button>
        </div>
      )}
    </div>
  );
}

export default CompanyList;
```

src/components/CompanyDetail.js

```
import React, { useState, useEffect } from 'react';
import { useParams } from 'react-router-dom';
import api from '../services/api';

function CompanyDetail() {
  const { symbol } = useParams();
  const [company, setCompany] = useState(null);
  const [fundamentals, setFundamentals] = useState(null);
  const [technicals, setTechnicals] = useState(null);
  const [dividends, setDividends] = useState([]);
  const [news, setNews] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [lastUpdated, setLastUpdated] = useState(new Date());

  useEffect(() => {
    const fetchCompanyData = async () => {
      try {
        setLoading(true);
        const response = await api.getCompanyDetails(symbol);
```

```
      if (response.data) {
        setCompany(response.data.company);
        setFundamentals(response.data.fundamentals);
        setTechnicals(response.data.technicals);
        setDividends(response.data.dividends || []);
        setNews(response.data.news || []);
        setLastUpdated(new Date());
      } else {
        setError('No data received from server');
      }

      setLoading(false);
    } catch (err) {
      setError(err.message);
      setLoading(false);
    }
  };

  fetchCompanyData();
}, [symbol]);

if (loading) return <div className="loading">Loading...</div>;
if (error) return <div className="error">Error: {error}</div>;
if (!company) return <div className="error">Company not found. The symbol {symbol} might not exist in the database.</div>;

return (
  <div className="company-detail">
    <div className="company-header">
      <h1>{company.Company_name || 'Unknown Company'}</h1>
      <p className="symbol">{company.Symbol}</p>
      <p className="sector">{company.Sector || 'N/A'}</p>
      <p className="last-updated">Last Updated: {lastUpdated.toLocaleTimeString()}</p>
    </div>

    <div className="detail-grid">
      {/* Company Profile */}
      <div className="detail-card">
        <h2>Company Profile</h2>
        <div className="data-grid">
          <div className="data-item">
            <label>Market Cap</label>
            <value>{company.Market_cap ? company.Market_cap.toLocaleString() : 'N/A'}</value>
          </div>
          <div className="data-item">
            <label>Paid Up Capital</label>
            <value>{company.Paidup_capital ? company.Paidup_capital.toLocaleString() : 'N/A'}</value>
          </div>
          <div className="data-item">
            <label>Listed Shares</label>
            <value>{company.Listed_share ? company.Listed_share.toLocaleString() : 'N/A'}</value>
          </div>
        </div>
      </div>

      {/* Technical Signals */}
      {technicals && (
        <div className="detail-card">
          <h2>Technical Indicators</h2>
          <div className="data-grid">
            <div className="data-item">
              <label>ADX</label>
              <value>{technicals.ADX || 'N/A'}</value>
            </div>
            <div className="data-item">
              <label>RSI</label>
              <value>{technicals.RSI || 'N/A'}</value>
            </div>
            <div className="data-item">
              <label>MACD</label>
              <value>{technicals.MACD || 'N/A'}</value>
            </div>
            <div className="data-item">
              <label>LTP</label>
              <value>{technicals.LTP || 'N/A'}</value>
            </div>
          </div>
        </div>
      )}

      {/* Fundamental Data */}
      {fundamentals && (
        <div className="detail-card">
          <h2>Fundamental Data</h2>
          <div className="data-grid">
            <div className="data-item">
              <label>Book Value</label>
              <value>{fundamentals.Book_Value || 'N/A'}</value>
            </div>
            <div className="data-item">
              <label>ROE</label>
              <value>{fundamentals.ROE ? `${fundamentals.ROE}%` : 'N/A'}</value>
            </div>
            <div className="data-item">
              <label>P/E Ratio</label>
              <value>{fundamentals.PE_Ratio || 'N/A'}</value>
            </div>
            <div className="data-item">
              <label>EPS</label>
              <value>{fundamentals.EPS || 'N/A'}</value>
            </div>
          </div>
        </div>
      )}
```

```
        {/* Dividend History */}
        {dividends && dividends.length > 0 && (
          <div className="detail-card">
            <h2>Dividend History</h2>
            <div className="dividend-table">
              <table>
                <thead>
                  <tr>
                    <th>Fiscal Year</th>
                    <th>Bonus Dividend</th>
                    <th>Cash Dividend</th>
                  </tr>
                </thead>
                <tbody>
                  {dividends.map((dividend, index) => (
                    <tr key={index}>
                      <td>{dividend.fiscal_year}</td>
                      <td>{dividend.bonus_dividend || '0'}%</td>
                      <td>{dividend.cash_dividend || '0'}%</td>
                    </tr>
                  ))}
                </tbody>
              </table>
            </div>
          </div>
        )}

        {/* Latest News */}
        {news && news.length > 0 && (
          <div className="detail-card full-width">
            <h2>Latest News</h2>
            <div className="news-list">
              {news.map((item, index) => (
                <div key={index} className="news-item">
                  <h3>{item.title}</h3>
                  <p className="news-date">{new Date(item.date).toLocaleDateString()}</p>
                  <p>{item.summary}</p>
                  {item.url && (
                    <a href={item.url} target="_blank" rel="noopener noreferrer" className="news-link">
                      Read more
                    </a>
                  )}
                </div>
              ))}
            </div>
          </div>
        )}
      </div>
    </div>
  );
}

export default CompanyDetail;
```

src/components/Dashboard.js

```
import React, { useState, useEffect } from 'react';
import { Link } from 'react-router-dom';
import api from '../services/api';

function Dashboard({ user }) {
  const [holdings, setHoldings] = useState([]);
  const [watchlist, setWatchlist] = useState([]);
  const [transactions, setTransactions] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [lastUpdated, setLastUpdated] = useState(new Date());
  const [showTransactionForm, setShowTransactionForm] = useState(false);
  const [transactionData, setTransactionData] = useState({
    symbol: '',
    quantity: '',
    rate: '',
    buy_sell: 'buy'
  });

  // Function to fetch all user data
  useEffect(() => {
    const fetchUserData = async () => {
      try {
        setLoading(true);

        const [portfolioRes, watchlistRes, transactionsRes] = await Promise.all([
          api.getPortfolio(),
          api.getWatchlist(),
          api.getTransactions()
        ]);

        setHoldings(portfolioRes.data || []);
        setWatchlist(watchlistRes.data || []);
        setTransactions(transactionsRes.data || []);
        setLastUpdated(new Date());
        setLoading(false);
      } catch (err) {
        console.error('Dashboard data fetch error:', err);
        setError(err.message || 'Failed to load dashboard data');
        setLoading(false);
      }
    };

    fetchUserData();
  }, []);

  const handleTransactionSubmit = async (e) => {
    e.preventDefault();
```

```
  try {
    await api.addTransaction(transactionData);

    // Refresh portfolio and transactions data
    const [portfolioRes, transactionsRes] = await Promise.all([
      api.getPortfolio(),
      api.getTransactions()
    ]);

    setHoldings(portfolioRes.data);
    setTransactions(transactionsRes.data);
    setShowTransactionForm(false);
    setTransactionData({
      symbol: '',
      quantity: '',
      rate: '',
      buy_sell: 'buy'
    });
  } catch (err) {
    setError(err.message || 'Failed to add transaction');
  }
};

const handleRemoveFromWatchlist = async (symbol) => {
  try {
    await api.removeFromWatchlist(symbol);
    setWatchlist(watchlist.filter(item => item.Symbol !== symbol));
  } catch (err) {
    setError(err.message || 'Failed to remove from watchlist');
  }
};

if (loading) return <div className="loading">Loading dashboard...</div>;
if (error) return <div className="error">Error: {error}</div>;

// Calculate total portfolio value
const portfolioValue = holdings.reduce((total, holding) => {
  return total + (holding.quantity * holding.last_price);
}, 0);

return (
  <div className="dashboard">
    <h1>Dashboard</h1>
    <p className="last-updated">Last Updated: {lastUpdated.toLocaleTimeString()}</p>

    {/* Portfolio Summary */}
    <div className="dashboard-section">
      <div className="section-header">
        <h2>Portfolio Summary</h2>
      </div>
      <div className="portfolio-summary">
        <div className="summary-card">
          <div className="summary-value">${portfolioValue.toLocaleString()}</div>
          <div className="summary-label">Total Value</div>
        </div>
        <div className="summary-card">
          <div className="summary-value">{holdings.length}</div>
          <div className="summary-label">Companies</div>
        </div>
        <div className="summary-card">
          <div className="summary-value">{transactions.length}</div>
          <div className="summary-label">Transactions</div>
        </div>
      </div>
    </div>

    {/* Holdings */}
    <div className="dashboard-section">
      <div className="section-header">
        <h2>Your Holdings</h2>
        <button
          className="add-transaction-btn"
          onClick={() => setShowTransactionForm(true)}
        >
          Add Transaction
        </button>
      </div>

      {showTransactionForm && (
        <div className="transaction-form-container">
          <form onSubmit={handleTransactionSubmit} className="transaction-form">
            <h3>Add Transaction</h3>
            <div className="form-group">
              <label>Symbol:</label>
              <input
                type="text"
                value={transactionData.symbol}
                onChange={(e) => setTransactionData({
                  ...transactionData,
                  symbol: e.target.value.toUpperCase()
                })}
                required
              />
            </div>
            <div className="form-group">
              <label>Quantity:</label>
              <input
                type="number"
                value={transactionData.quantity}
                onChange={(e) => setTransactionData({
                  ...transactionData,
                  quantity: e.target.value
                })}
```

```
            required
          />
        </div>
        <div className="form-group">
          <label>Rate:</label>
          <input
            type="number"
            step="0.01"
            value={transactionData.rate}
            onChange={(e) => setTransactionData({
              ...transactionData,
              rate: e.target.value
            })}
            required
          />
        </div>
        <div className="form-group">
          <label>Type:</label>
          <select
            value={transactionData.buy_sell}
            onChange={(e) => setTransactionData({
              ...transactionData,
              buy_sell: e.target.value
            })}
          >
            <option value="buy">Buy</option>
            <option value="sell">Sell</option>
          </select>
        </div>
        <div className="form-buttons">
          <button type="submit">Add Transaction</button>
          <button
            type="button"
            className="cancel-btn"
            onClick={() => setShowTransactionForm(false)}
          >
            Cancel
          </button>
        </div>
      </form>
    </div>
  )}

  {holdings.length > 0 ? (
    <div className="holdings-table">
      <table>
        <thead>
          <tr>
            <th>Symbol</th>
            <th>Company</th>
            <th>Quantity</th>
            <th>Avg. Buy</th>
            <th>Current Price</th>
            <th>Value</th>
            <th>Gain/Loss</th>
          </tr>
        </thead>
        <tbody>
          {holdings.map((holding) => {
            const value = holding.quantity * holding.last_price;
            const invested = holding.quantity * holding.avg_buy_price;
            const gainLoss = value - invested;
            const gainLossPercent = (gainLoss / invested) * 100;

            return (
              <tr key={holding.symbol}>
                <td>
                  <Link to={`/company/${holding.symbol}`}>{holding.symbol}</Link>
                </td>
                <td>{holding.company_name}</td>
                <td>{holding.quantity}</td>
                <td>${holding.avg_buy_price.toFixed(2)}</td>
                <td>${holding.last_price.toFixed(2)}</td>
                <td>${value.toFixed(2)}</td>
                <td className={gainLoss >= 0 ? 'positive' : 'negative'}>
                  ${Math.abs(gainLoss).toFixed(2)}
                  ({gainLossPercent >= 0 ? '+' : '-'}{Math.abs(gainLossPercent).toFixed(2)}%)
                </td>
              </tr>
            );
          })}
        </tbody>
      </table>
    </div>
  ) : (
    <div className="empty-state">
      <p>You don't have any holdings yet.</p>
      <button
        onClick={() => setShowTransactionForm(true)}
        className="action-btn"
      >
        Add Your First Investment
      </button>
    </div>
  )}
</div>

{/* Watchlist */}
<div className="dashboard-section">
  <div className="section-header">
    <h2>Your Watchlist</h2>
    <Link to="/companies" className="view-all-btn">View All Companies</Link>
  </div>
```

```
        {watchlist.length > 0 ? (
          <div className="watchlist-grid">
            {watchlist.map((item) => (
              <div key={item.Symbol} className="watchlist-card">
                <div className="watchlist-header">
                  <Link to={`/company/${item.Symbol}`}>
                    <h3>{item.Company_name}</h3>
                    <span className="symbol">{item.Symbol}</span>
                  </Link>
                  <button
                    className="remove-btn"
                    onClick={() => handleRemoveFromWatchlist(item.Symbol)}
                  >
                    ×
                  </button>
                </div>
                <div className="watchlist-data">
                  <div className="data-row">
                    <span className="label">Price:</span>
                    <span className="value">${item.Last_price}</span>
                  </div>
                  <div className="data-row">
                    <span className="label">Change:</span>
                    <span className={`value ${item.Change >= 0 ? 'positive' : 'negative'}`}>
                      {item.Change >= 0 ? '+' : ''}{item.Change}%
                    </span>
                  </div>
                </div>
              </div>
            ))}
          </div>
        ) : (
          <div className="empty-state">
            <p>You don't have any companies in your watchlist.</p>
            <Link to="/companies" className="action-btn">
              Browse Companies
            </Link>
          </div>
        )}
      </div>

      {/* Recent Transactions */}
      <div className="dashboard-section">
        <div className="section-header">
          <h2>Recent Transactions</h2>
        </div>

        {transactions.length > 0 ? (
          <div className="transactions-table">
            <table>
              <thead>
                <tr>
                  <th>Date</th>
                  <th>Symbol</th>
                  <th>Type</th>
                  <th>Quantity</th>
                  <th>Price</th>
                  <th>Total</th>
                </tr>
              </thead>
              <tbody>
                {transactions.slice(0, 5).map((transaction, index) => {
                  const total = transaction.quantity * transaction.rate;

                  return (
                    <tr key={index}>
                      <td>{new Date(transaction.date).toLocaleDateString()}</td>
                      <td>
                        <Link to={`/company/${transaction.symbol}`}>{transaction.symbol}</Link>
                      </td>
                      <td className={transaction.buy_sell === 'buy' ? 'buy' : 'sell'}>
                        {transaction.buy_sell === 'buy' ? 'Buy' : 'Sell'}
                      </td>
                      <td>{transaction.quantity}</td>
                      <td>${transaction.rate.toFixed(2)}</td>
                      <td>${total.toFixed(2)}</td>
                    </tr>
                  );
                })}
              </tbody>
            </table>
          </div>
        ) : (
          <div className="empty-state">
            <p>You haven't made any transactions yet.</p>
            <button
              onClick={() => setShowTransactionForm(true)}
              className="action-btn"
            >
              Record Your First Transaction
            </button>
          </div>
        )}
      </div>
    </div>
  );
}

export default Dashboard;
```