PRAKTIK PER. 7

Langkah 1: Setup Project dan Dependencies

```
npm create vite@latest praktik-7 -- --template react
```

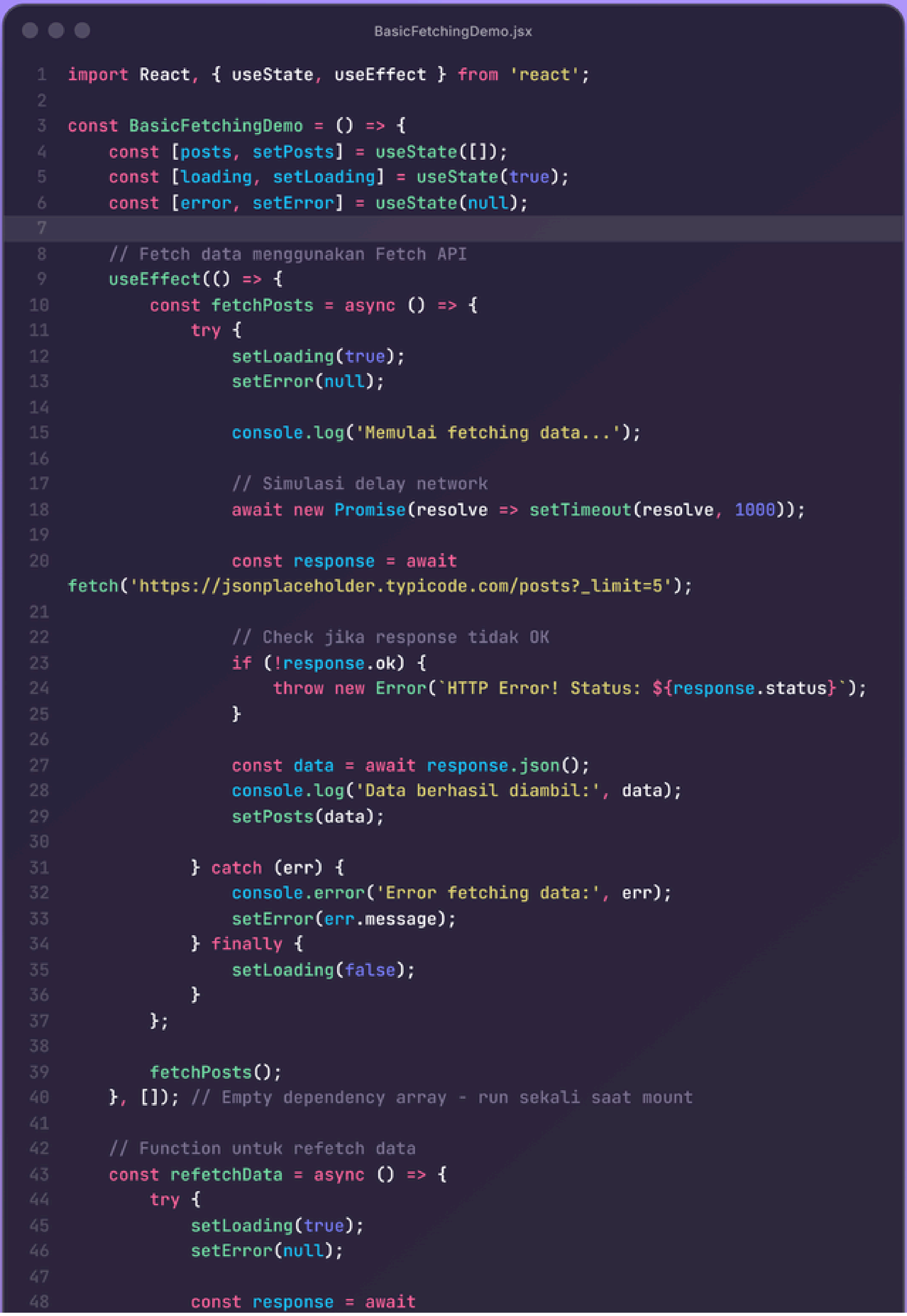
```
npm install
```

```
npm install axios
```

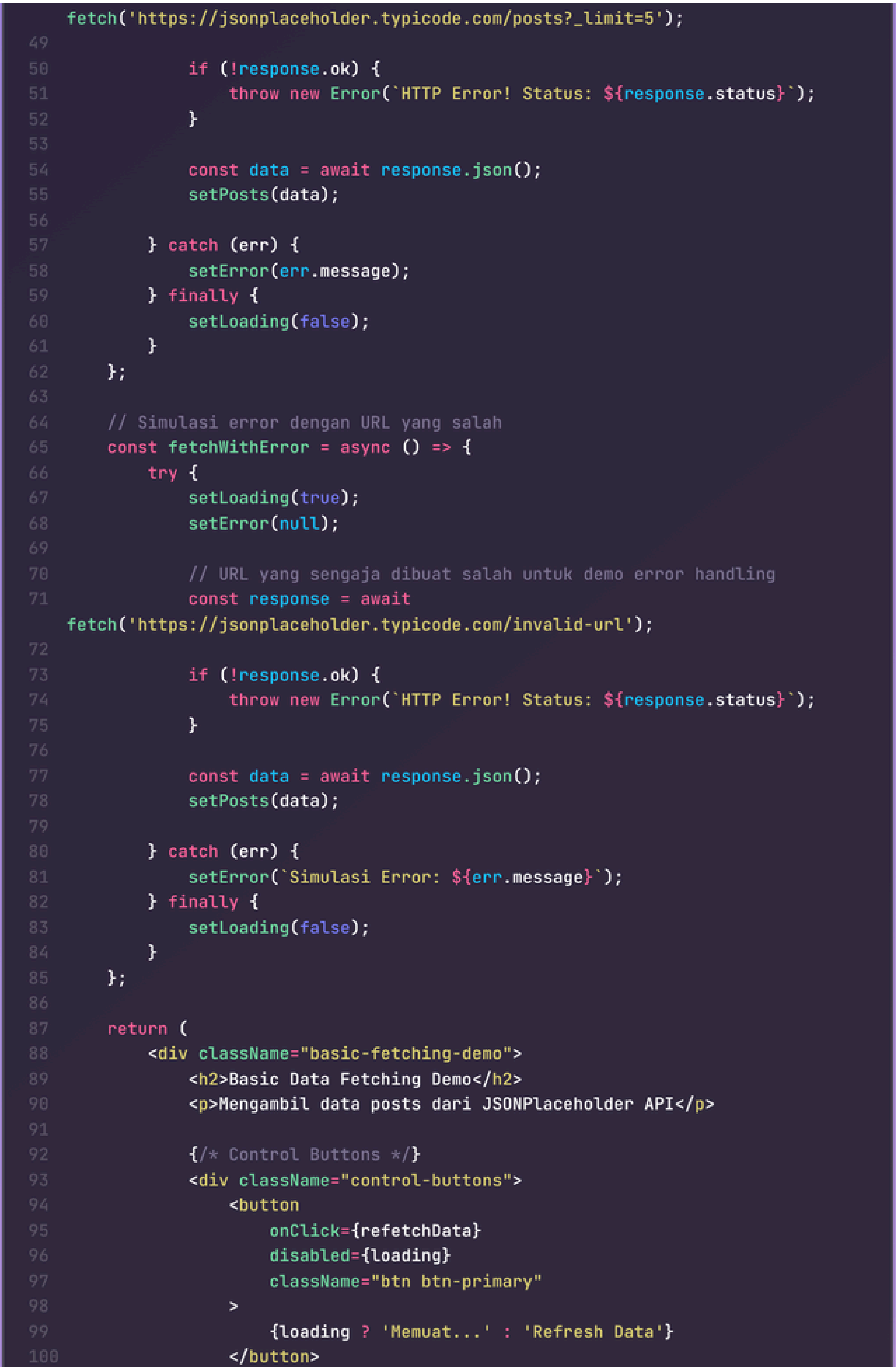
Langkah 2: Komponen Basic Fetching Demo

File : src/components/BasicFetchingDemo.jsx

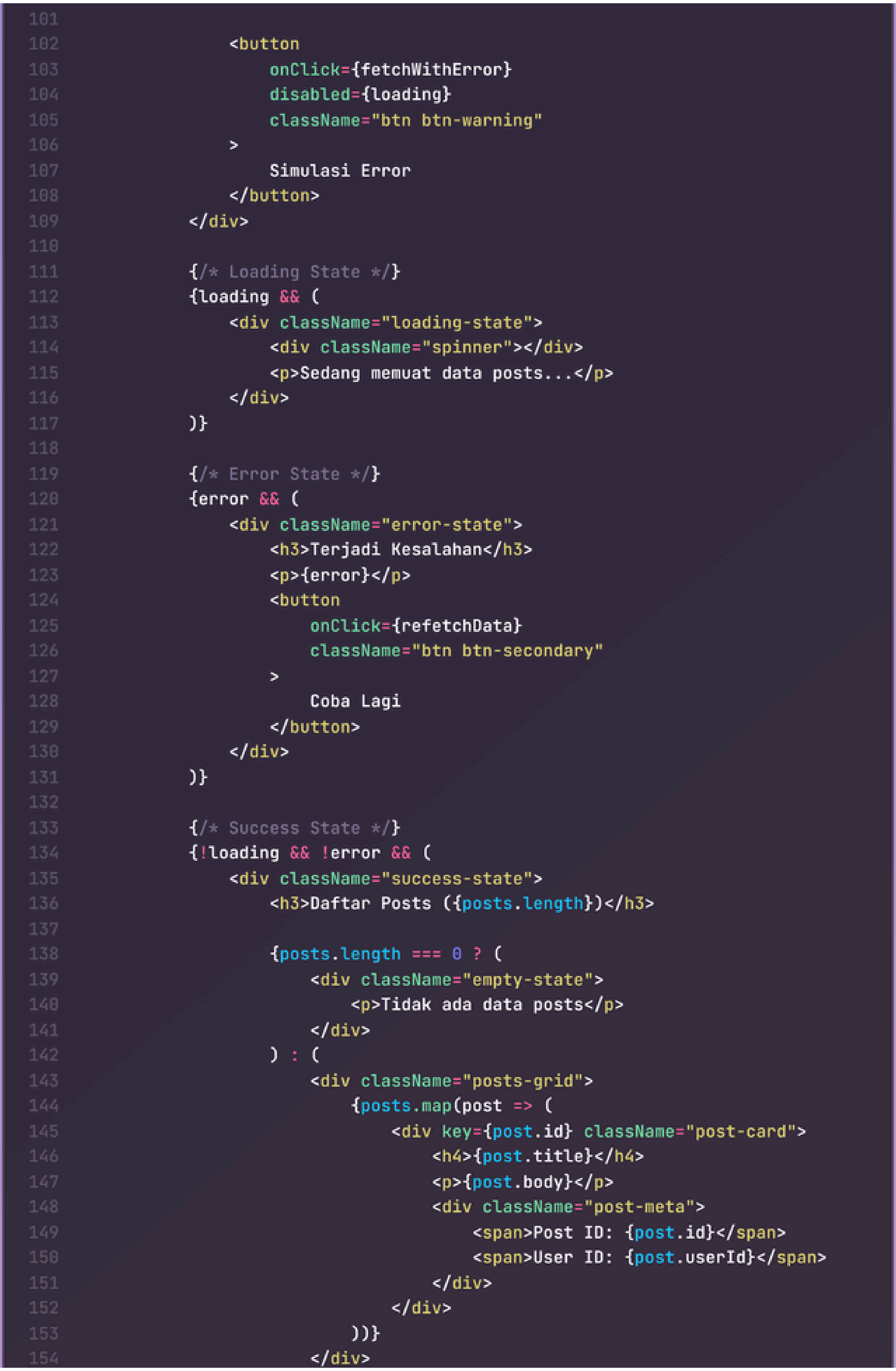
Penjelasan: Komponen ini menunjukkan dasar-dasar data fetching dengan Fetch API. Kita akan mengambil data posts dari JSONPlaceholder API dan menampilkannya dengan handling state yang proper.


BasicFetchingDemo.jsx

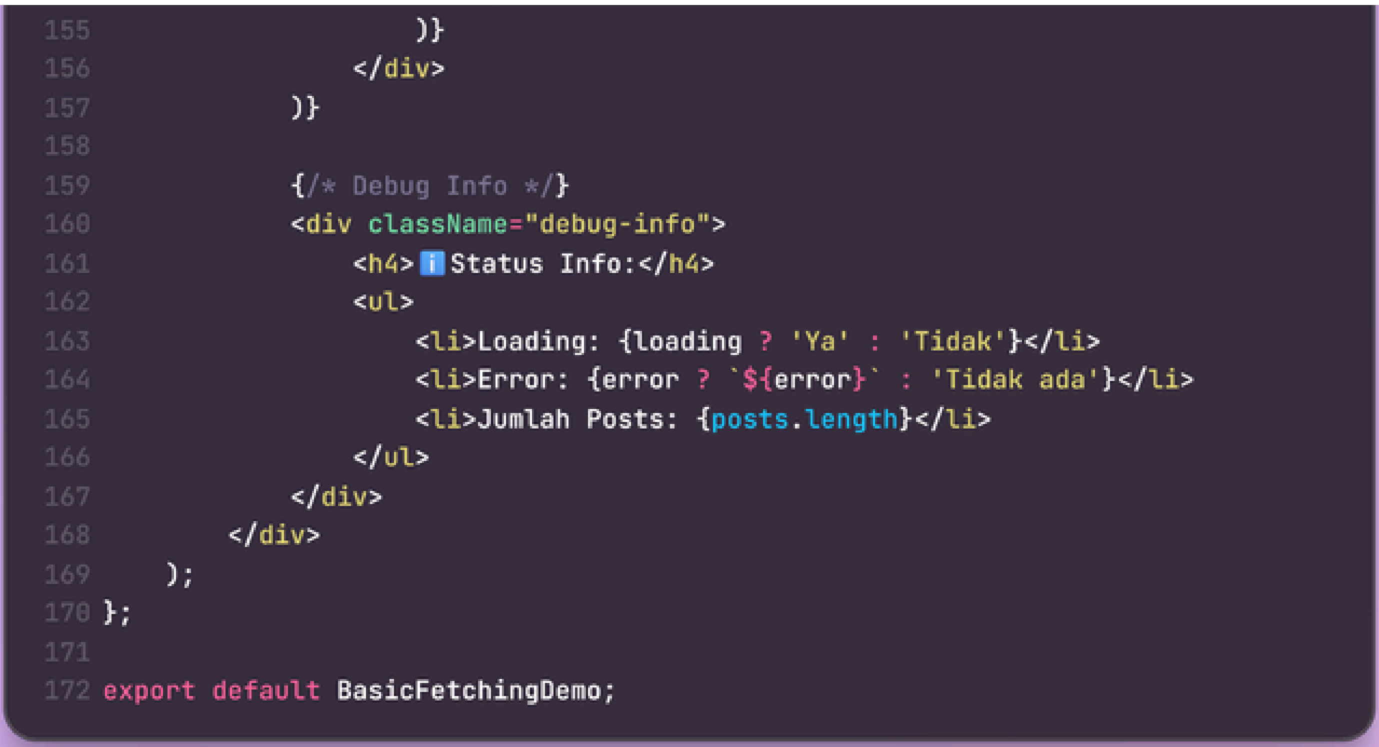
```
import React, { useState, useEffect } from 'react';

const BasicFetchingDemo = () => {
    const [posts, setPosts] = useState([]);
    const [loading, setLoading] = useState(true);
    const [error, setError] = useState(null);

    // Fetch data menggunakan Fetch API
    useEffect(() => {
        const fetchPosts = async () => {
            try {
                setLoading(true);
                setError(null);

                console.log('Memulai fetching data...');

                // Simulasi delay network
                await new Promise(resolve => setTimeout(resolve, 1000));

                const response = await
fetch('https://jsonplaceholder.typicode.com/posts?_limit=5');

                // Check jika response tidak OK
                if (!response.ok) {
                    throw new Error(`HTTP Error! Status: ${response.status}`);
                }

                const data = await response.json();
                console.log('Data berhasil diambil:', data);
                setPosts(data);

            } catch (err) {
                console.error('Error fetching data:', err);
                setError(err.message);
            } finally {
                setLoading(false);
            }
        };

        fetchPosts();
    }, []); // Empty dependency array - run sekali saat mount

    // Function untuk refetch data
    const refetchData = async () => {
        try {
            setLoading(true);
            setError(null);

            const response = await
```



```
fetch('https://jsonplaceholder.typicode.com/posts?_limit=5');

            if (!response.ok) {
                throw new Error(`HTTP Error! Status: ${response.status}`);
            }

            const data = await response.json();
            setPosts(data);

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

    // Simulasi error dengan URL yang salah
    const fetchWithError = async () => {
        try {
            setLoading(true);
            setError(null);

            // URL yang sengaja dibuat salah untuk demo error handling
            const response = await
fetch('https://jsonplaceholder.typicode.com/invalid-url');

            if (!response.ok) {
                throw new Error(`HTTP Error! Status: ${response.status}`);
            }

            const data = await response.json();
            setPosts(data);

        } catch (err) {
            setError(`Simulasi Error: ${err.message}`);
        } finally {
            setLoading(false);
        }
    };

    return (
        <div className="basic-fetching-demo">
            <h2>Basic Data Fetching Demo</h2>
            <p>Mengambil data posts dari JSONPlaceholder API</p>

            {/* Control Buttons */}
            <div className="control-buttons">
                <button
                    onClick={refetchData}
                    disabled={loading}
                    className="btn btn-primary"
                >
                    {loading ? 'Memuat...' : 'Refresh Data'}
                </button>
```



```jsx

                <button
                    onClick={fetchWithError}
                    disabled={loading}
                    className="btn btn-warning"
                >
                    Simulasi Error
                </button>
            </div>

            {/* Loading State */}
            {loading && (
                <div className="loading-state">
                    <div className="spinner"></div>
                    <p>Sedang memuat data posts...</p>
                </div>
            )}

            {/* Error State */}
            {error && (
                <div className="error-state">
                    <h3>Terjadi Kesalahan</h3>
                    <p>{error}</p>
                    <button
                        onClick={refetchData}
                        className="btn btn-secondary"
                    >
                        Coba Lagi
                    </button>
                </div>
            )}

            {/* Success State */}
            {!loading && !error && (
                <div className="success-state">
                    <h3>Daftar Posts ({posts.length})</h3>

                    {posts.length === 0 ? (
                        <div className="empty-state">
                            <p>Tidak ada data posts</p>
                        </div>
                    ) : (
                        <div className="posts-grid">
                            {posts.map(post => (
                                <div key={post.id} className="post-card">
                                    <h4>{post.title}</h4>
                                    <p>{post.body}</p>
                                    <div className="post-meta">
                                        <span>Post ID: {post.id}</span>
                                        <span>User ID: {post.userId}</span>
                                    </div>
                                </div>
                            ))}
                        </div>
```

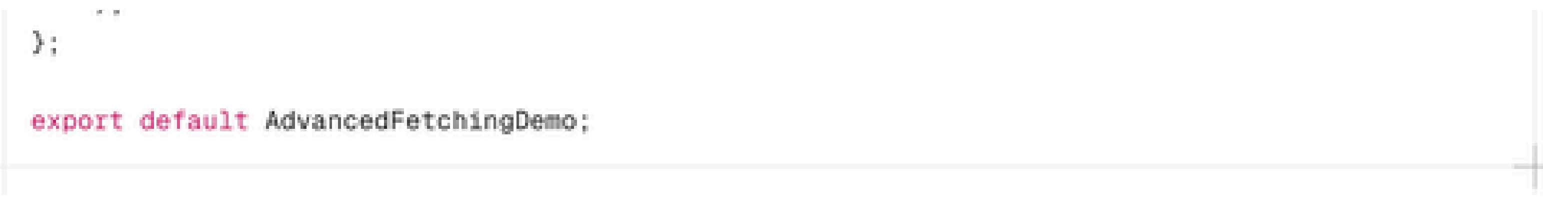


```
                    )}
                </div>
            )}

            {/* Debug Info */}
            <div className="debug-info">
                <h4>ℹ️ Status Info:</h4>
                <ul>
                    <li>Loading: {loading ? 'Ya' : 'Tidak'}</li>
                    <li>Error: {error ? `${error}` : 'Tidak ada'}</li>
                    <li>Jumlah Posts: {posts.length}</li>
                </ul>
            </div>
        </div>
    );
};

export default BasicFetchingDemo;
```

Langkah 3: Komponen Advanced Fetching Demo

File : src/components/AdvancedFetchingDemo.jsx

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';

const AdvancedFetchingDemo = () => {
    const [users, setUsers] = useState([]);
    const [selectedUserId, setSelectedUserId] = useState('');
    const [userPosts, setUserPosts] = useState([]);
    const [loading, setLoading] = useState({
        users: true,
        posts: false
    });
    const [error, setError] = useState(null);
    const [searchTerm, setSearchTerm] = useState('');

    // Fetch semua users menggunakan Axios
    useEffect(() => {
        const fetchUsers = async () => {
            try {
                setLoading(prev => ({ ...prev, users: true }));
                setError(null);

                console.log('Fetching users...');

                // Axios otomatis handle JSON parsing dan error status
                const response = await
axios.get('https://jsonplaceholder.typicode.com/users');

                console.log('Users berhasil diambil:', response.data);
                setUsers(response.data);

            } catch (err) {
                console.error('Error fetching users:', err);
                setError(`Gagal mengambil data users: ${err.message}`);
            } finally {
                setLoading(prev => ({ ...prev, users: false }));
            }
        };

        fetchUsers();
    }, []);

    // Fetch posts berdasarkan user ID (dependent fetching)
    useEffect(() => {
        if (!selectedUserId) return;

        const fetchUserPosts = async () => {
            try {
                setLoading(prev => ({ ...prev, posts: true }));
                setError(null);

                console.log(`Fetching posts untuk user ${selectedUserId}...`);

                // Menggunakan Axios dengan error handling built-in
                // Menggunakan Axios dengan error handling built-in
                const response = await axios.get(
                    `https://jsonplaceholder.typicode.com/posts?userId=${selectedUserId}`
                );

                console.log('Posts berhasil diambil:', response.data);
                setUserPosts(response.data);

            } catch (err) {
                console.error('Error fetching posts:', err);
                setError(`Gagal mengambil posts: ${err.message}`);
            } finally {
                setLoading(prev => ({ ...prev, posts: false }));
            }
        };

        fetchUserPosts();
    }, [selectedUserId]); // Dependency: re-fetch ketika selectedUserId berubah

    // Filter users berdasarkan search term
    const filteredUsers = users.filter(user =>
        user.name.toLowerCase().includes(searchTerm.toLowerCase()) ||
        user.email.toLowerCase().includes(searchTerm.toLowerCase())
    );

    // Reset selection
    const resetSelection = () => {
        setSelectedUserId('');
        setUserPosts([]);
        setSearchTerm('');
    };
    return (
        <div className="advanced-fetching-demo">
            <h2>Advanced Data Fetching Demo</h2>
            <p>Dependent fetching, search, dan optimisasi dengan Axios</p>

            {/* Search Box */}
            <div className="search-section">
                <input
                    type="text"
                    placeholder="Cari user berdasarkan nama atau email..."
                    value={searchTerm}
                    onChange={(e) => setSearchTerm(e.target.value)}
                    className="search-input"
                />
                <button onClick={resetSelection} className="btn btn-secondary">
                    Reset
                </button>
            </div>

            {/* Users List */}
            <div className="users-section">
                <h3>Daftar Users</h3>

                {loading.users ? (
                    <div className="loading-state">
                        <div className="spinner"></div>
                        <p>Memuat daftar users...</p>
                    </div>
                ) : error ? (
                    <div className="error-state">
                        <p>{error}</p>
                    </div>
                ) : (
                    <div className="users-grid">
                        {filteredUsers.map(user => (
                            <div
                                key={user.id}
                                className={`user-card ${selectedUserId === user.id.toString()
? 'active' : ''}`}
                                onClick={() => setSelectedUserId(user.id.toString())}
                            >
                                <h4>{user.name}</h4>
                                <p>{user.email}</p>
                                <p>{user.company.name}</p>
                                <p>{user.website}</p>
                            </div>
                        ))}
                    </div>
                )}
            </div>

            {/* User Posts */}
            {selectedUserId && (
                <div className="posts-section">
                    <h3>
                        Posts dari User {users.find(u => u.id.toString() ===
selectedUserId)?.name}
                        {loading.posts && <span className="loading-badge">Loading...</span>}
                    </h3>

                    {error && !loading.posts ? (
                        <div className="error-state">
                            <p>{error}</p>
                        </div>
                    ) : (
                        <div className="posts-list">
                            {userPosts.map(post => (
                                <div key={post.id} className="post-item">
                                    <h4>{post.title}</h4>
                                    <p>{post.body}</p>
                                </div>
                            ))}

                            {userPosts.length === 0 && !loading.posts && (
                                <div className="empty-state">
                                    <p>User ini belum membuat posts</p>
                                </div>
                            )}
                        </div>
                    )}
                </div>
            )}

            {/* Statistics */}
            <div className="stats-section">
                <h4>Statistics:</h4>
                <div className="stats-grid">
                    <div className="stat-card">
                        <span className="stat-number">{users.length}</span>
                        <span className="stat-label">Total Users</span>
                    </div>
                    <div className="stat-card">
                        <span className="stat-number">{filteredUsers.length}</span>
                        <span className="stat-label">Filtered Users</span>
                    </div>
                    <div className="stat-card">
                        <span className="stat-number">{userPosts.length}</span>
                        <span className="stat-label">User Posts</span>
                    </div>
                    <div className="stat-card">
                        <span className="stat-number">{selectedUserId ? 'Ya' : 'Tidak'}</span>
                        <span className="stat-label">User Selected</span>
                    </div>
                </div>
            </div>
        </div>
    );
```



```
};

export default AdvancedFetchingDemo;
```

Langkah 4: Komponen CRUD Operations Demo

File : src/components/CRUDOperationsDemo.jsx

```
import React, { useState, useEffect } from 'react';

const CRUDOperationsDemo = () => {
    const [todos, setTodos] = useState([]);
    const [loading, setLoading] = useState(false);
    const [error, setError] = useState(null);
    const [editingTodo, setEditingTodo] = useState(null);
    const [formData, setFormData] = useState({
        title: '',
        completed: false
    });

    // Fetch todos
    const fetchTodos = async () => {
        try {
            setLoading(true);
            setError(null);

            const response = await fetch('https://jsonplaceholder.typicode.com/todos?
_limit=5');

            if (!response.ok) {
                throw new Error('Failed to fetch todos');
            }

            const data = await response.json();
            setTodos(data);

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

    // Initial fetch
    useEffect(() => {
        fetchTodos();
    }, []);

    // Create new todo
    const createTodo = async (e) => {
        e.preventDefault();

        try {
            setLoading(true);
            setError(null);

            const response = await fetch('https://jsonplaceholder.typicode.com/todos', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
                },
            },
            body: JSON.stringify({
                title: formData.title,
                completed: formData.completed,
                userId: 1
            })
        });

        if (!response.ok) {
            throw new Error('Failed to create todo');
        }

        const newTodo = await response.json();

        // JSONPlaceholder tidak benar-benar menyimpan data, jadi kita simulasi
        newTodo.id = Date.now(); // ID sementara
        setTodos(prev => [newTodo, ...prev]);

        // Reset form
        setFormData({ title: '', completed: false });

        console.log('Todo created:', newTodo);

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

// Update todo
const updateTodo = async (e) => {
    e.preventDefault();

    try {
        setLoading(true);
        setError(null);

        const response = await
fetch(`https://jsonplaceholder.typicode.com/todos/${editingTodo.id}`, {
            method: 'PUT',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({
                ...editingTodo,
                title: formData.title,
                completed: formData.completed
            })
        });

        if (!response.ok) {
            throw new Error('Failed to update todo');
        }
        const updatedTodo = await response.json();

        // Update local state
        setTodos(prev => prev.map(todo =>
            todo.id === editingTodo.id ? { ...todo, ...updatedTodo } : todo
        ));

        // Reset editing state
        setEditingTodo(null);
        setFormData({ title: '', completed: false });

        console.log('Todo updated:', updatedTodo);

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

// Delete todo
const deleteTodo = async (id) => {
    try {
        setLoading(true);
        setError(null);

        const response = await fetch(`https://jsonplaceholder.typicode.com/todos/${id}`, {
            method: 'DELETE'
        });

        if (!response.ok) {
            throw new Error('Failed to delete todo');
        }

        // Remove from local state
        setTodos(prev => prev.filter(todo => todo.id !== id));

        console.log('Todo deleted:', id);

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

// Start editing
const startEditing = (todo) => {
    setEditingTodo(todo);
    setFormData({
        title: todo.title,
        completed: todo.completed
    });
};
```

```
};

// Cancel editing
const cancelEditing = () => {
    setEditingTodo(null);
    setFormData({ title: '', completed: false });
};

// Handle form input changes
const handleInputChange = (e) => {
    const { name, value, type, checked } = e.target;
    setFormData(prev => ({
        ...prev,
        [name]: type === 'checkbox' ? checked : value
    }));
};
```

```
    return (
        <div className="crud-demo">
            <h2>CRUD Operations Demo</h2>
            <p>Create, Read, Update, Delete operations dengan REST API</p>

            {/* Todo Form */}
            <div className="todo-form-section">
                <h3>{editingTodo ? 'Edit Todo' : 'Add New Todo'}</h3>

                <form onSubmit={editingTodo ? updateTodo : createTodo} className="todo-form">
                    <div className="form-group">
                        <input
                            type="text"
                            name="title"
                            value={formData.title}
                            onChange={handleInputChange}
                            placeholder="Apa yang perlu dilakukan?"
                            required
                            className="form-input"
                        />
                    </div>

                    <div className="form-group checkbox-group">
                        <label>
                            <input
                                type="checkbox"
                                name="completed"
                                checked={formData.completed}
                                onChange={handleInputChange}
                            />
                            Selesai
                        </label>
                    </div>

                    <div className="form-buttons">
                        <button
                            type="submit"
                            disabled={loading}
                            className="btn btn-primary"
                        >
                            {loading ? 'Loading...' : editingTodo ? 'Update Todo' : 'Add
Todo'}
                        </button>

                        {editingTodo && (
                            <button
                                type="button"
                                onClick={cancelEditing}
                                className="btn btn-secondary"
                            >
                                Cancel
                            </button>
                        )}
                    </div>
                </form>
            </div>

            {/* Todos List */}
            <div className="todos-section">
                <div className="section-header">
                    <h3>Daftar Todos ({todos.length})</h3>
                    <button
                        onClick={fetchTodos}
                        disabled={loading}
                        className="btn btn-secondary"
                    >
                        Refresh
                    </button>
                </div>

                {loading && !editingTodo ? (
                    <div className="loading-state">
                        <div className="spinner"></div>
                        <p>Memuat todos...</p>
                    </div>
                ) : error ? (
                    <div className="error-state">
                        <p>{error}</p>
                        <button onClick={fetchTodos} className="btn btn-secondary">
                            Coba Lagi
                        </button>
                    </div>
                ) : (
                    <div className="todos-list">
                        {todos.map(todo => (
                            <div key={todo.id} className={`todo-card ${todo.completed ?
'completed' : ''}`}>
                                <div className="todo-content">
                                    <h4>{todo.title}</h4>
                                    <p>Status: {todo.completed ? 'Selesai' : 'Belum selesai'}
</p>

                                    <p>ID: {todo.id}</p>
                                </div>

                                <div className="todo-actions">
                                    <button
                                        onClick={() => startEditing(todo)}
                                        className="btn btn-warning btn-sm"
                                    >
                                        Edit
                                    </button>

                                    <button
                                        onClick={() => deleteTodo(todo.id)}
                                        disabled={loading}
                                        className="btn btn-danger btn-sm"
                                    >
```

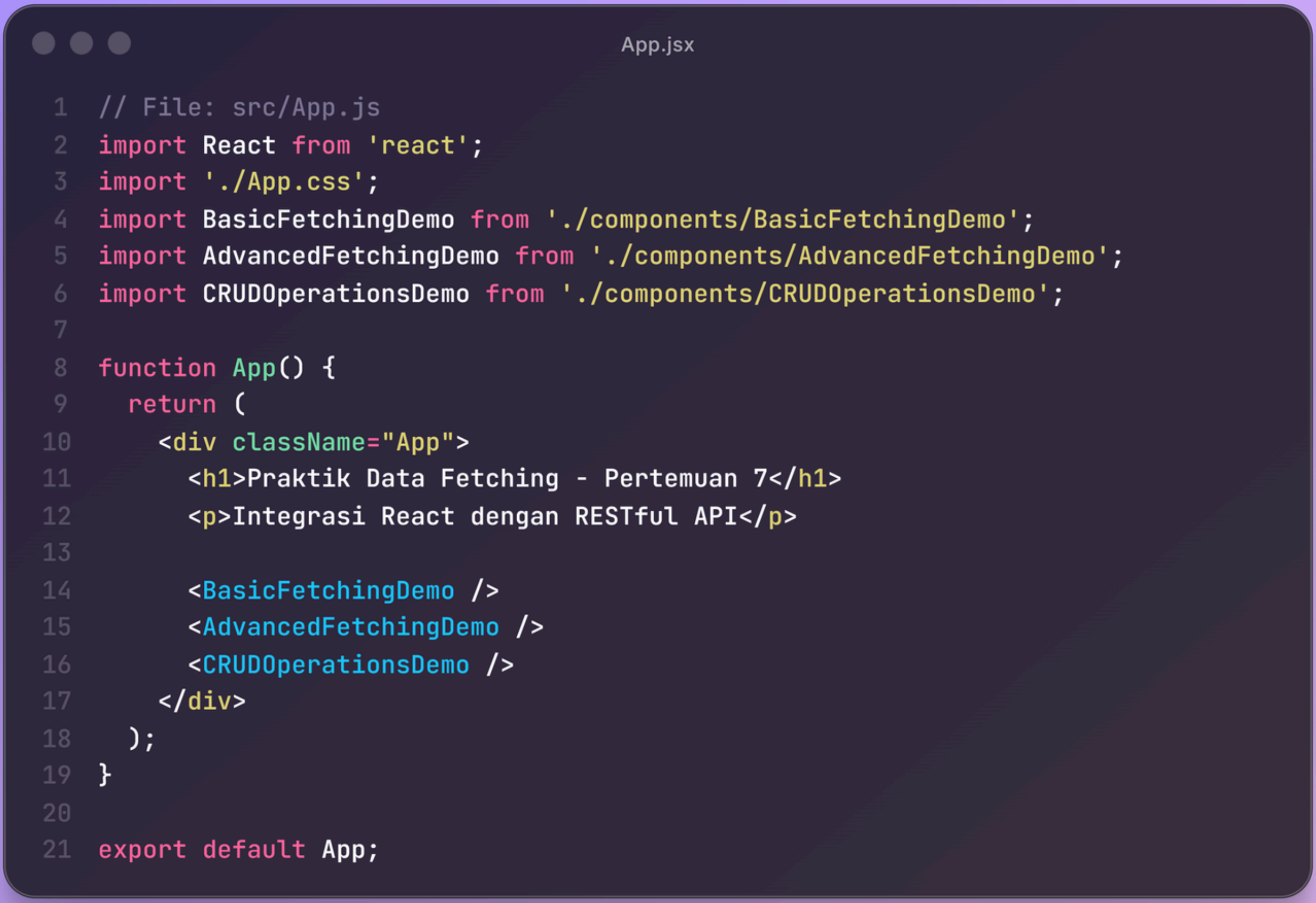
```
                                        Delete
                                    </button>
                                </div>
                            </div>
                        ))}

                        {todos.length === 0 && (
                            <div className="empty-state">
                                <p>Tidak ada todos. Buat yang pertama!</p>
                            </div>
                        )}
                    </div>
                )}
            </div>

            {/* API Info */}
            <div className="api-info">
                <h4>Tentang API:</h4>
                <p>
                    Menggunakan JSONPlaceholder - API dummy gratis.
                    <strong> Data tidak benar-benar disimpan</strong> di server,
                    tetapi kita bisa simulasi semua operasi CRUD.
                </p>
            </div>
        </div>
    );
};

export default CRUDOperationsDemo;
```

Langkah 5 : App.css

File : src/App.css

Langkah 1: Setup Project dan Dependencies

File : src/App.jsx


App.jsx

```jsx
// File: src/App.js
import React from 'react';
import './App.css';
import BasicFetchingDemo from './components/BasicFetchingDemo';
import AdvancedFetchingDemo from './components/AdvancedFetchingDemo';
import CRUDOperationsDemo from './components/CRUDOperationsDemo';

function App() {
  return (
    <div className="App">
      <h1>Praktik Data Fetching - Pertemuan 7</h1>
      <p>Integrasi React dengan RESTful API</p>

      <BasicFetchingDemo />
      <AdvancedFetchingDemo />
      <CRUDOperationsDemo />
    </div>
  );
}

export default App;
```