PRAKTIK PER. 6

Langkah 1: Setup Project

npm create vite@latest praktik-6 -- --template react

Langkah 2: Komponen Lifecycle Demo

File: src/components/LifecycleDemo.jsx

LifecycleDemo.jsx

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

const LifecycleDemo = () => {
    const [count, setCount] = useState(0);
    const [isVisible, setIsVisible] = useState(true);

    // Effect tanpa dependencies - run setelah setiap render
    useEffect(() => {
        console.log('Effect tanpa dependencies - dijalankan setelah setiap render');
    });

    // Effect dengan empty dependencies - run sekali setelah mount
    useEffect(() => {
        console.log('Component mounted - effect dengan []');

        return () => {
            console.log('Cleanup dari effect dengan [] - component akan unmount');
        };
    }, []);

    // Effect dengan dependency count - run ketika count berubah
    useEffect(() => {
        console.log('Count berubah:', count);

        // Update document title
        document.title = `Count: ${count}`;

        return () => {
            console.log('Cleanup sebelum count effect dijalankan lagi');
        };
    }, [count]);

    // Effect dengan dependency isVisible
    useEffect(() => {
        console.log('Visibility berubah:', isVisible);
    }, [isVisible]);

    const increment = () => {
        setCount(prev => prev + 1);
    };

    const toggleVisibility = () => {
        setIsVisible(prev => !prev);
    };

    return (
        <div className="lifecycle-demo">
```

```jsx
            <h2>Demo Lifecycle dengan useEffect</h2>

            <div className="counter-section">
                <h3>Counter: {count}</h3>
                <button onClick={increment} className="btn btn-primary">
                    Tambah Count
                </button>
            </div>

            <div className="visibility-section">
                <button onClick={toggleVisibility} className="btn btn-secondary">
                    {isVisible ? 'Sembunyikan' : 'Tampilkan'} Pesan
                </button>

                {isVisible && (
                    <div className="message-box">
                        <p>Pesan ini {isVisible ? 'terlihat' : 'tersembunyi'}</p>
                        <p>Count saat ini: {count}</p>
                    </div>
                )}
            </div>

            <div className="explanation">
                <h4>Perhatikan Console Browser:</h4>
                <p>Buka Developer Tools (F12) dan lihat console untuk melihat kapan effects dijalankan</p>
            </div>
        </div>
    );
};

export default LifecycleDemo;
```

Langkah 3: Komponen Data Fetching Demo

File: src/components/DataFetchingDemo.jsx

DataFetchingDemo.jsx

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

const DataFetchingDemo = () => {
    const [users, setUsers] = useState([]);
    const [loading, setLoading] = useState(true);
    const [error, setError] = useState(null);
    const [userId, setUserId] = useState(1);
    const [userDetail, setUserDetail] = useState(null);

    // Effect untuk fetching semua users (sekali saat mount)
    useEffect(() => {
        const fetchUsers = async () => {
            try {
                setLoading(true);
                setError(null);

                // Simulasi API call dengan timeout
                await new Promise(resolve => setTimeout(resolve, 1000));

                // Data dummy users
                const dummyUsers = [
                    { id: 1, name: 'John Doe', email: 'john@example.com' },
                    { id: 2, name: 'Jane Smith', email: 'jane@example.com' },
                    { id: 3, name: 'Bob Johnson', email: 'bob@example.com' },
                    { id: 4, name: 'Alice Brown', email: 'alice@example.com' }
                ];

                setUsers(dummyUsers);
            } catch (err) {
                setError('Gagal memuat data users');
                console.error('Error fetching users:', err);
            } finally {
                setLoading(false);
            }
        };

        fetchUsers();
    }, []); // Empty dependencies - hanya sekali saat mount

    // Effect untuk fetching user detail berdasarkan userId
    useEffect(() => {
        const fetchUserDetail = async () => {
            if (!userId) return;

            try {
                setLoading(true);

                // Simulasi API call
                await new Promise(resolve => setTimeout(resolve, 500));
```

```
                // Data dummy user detail
                const dummyUserDetail = {
                    id: userId,
                    name: `User ${userId}`,
                    email: `user${userId}@example.com`,
                    phone: `+62 812-3456-789${userId}`,
                    address: `Jl. Contoh No. ${userId}, Jakarta`
                };

                setUserDetail(dummyUserDetail);
            } catch (err) {
                setError(`Gagal memuat detail user ${userId}`);
                console.error('Error fetching user detail:', err);
            } finally {
                setLoading(false);
            }
        };

        fetchUserDetail();
    }, [userId]); // Dependency: userId - re-fetch ketika userId berubah

    const handleUserChange = (event) => {
        setUserId(parseInt(event.target.value));
    };

    return (
        <div className="data-fetching-demo">
            <h2>Demo Data Fetching dengan useEffect</h2>

            {/* Loading State */}
            {loading && (
                <div className="loading">
                    <div className="spinner"></div>
                    <p>Memuat data...</p>
                </div>
            )}

            {/* Error State */}
            {error && (
                <div className="error-message">
                    {error}
                </div>
            )}

            {/* Users List */}
            <div className="users-section">
                <h3>Daftar Users</h3>
                <div className="users-grid">
                    {users.map(user => (
                        <div
                            key={user.id}
                            className={`user-card ${userId === user.id ?
'active' : ''}`}
```

```
                            onClick={() => setUserId(user.id)}
                        >
                            <h4>{user.name}</h4>
                            <p>{user.email}</p>
                        </div>
                    ))}
                </div>
            </div>

            {/* User Detail */}
            <div className="user-detail-section">
                <h3>Detail User</h3>

                <div className="user-selector">
                    <label>Pilih User ID: </label>
                    <select value={userId} onChange={handleUserChange}>
                        {[1, 2, 3, 4].map(id => (
                            <option key={id} value={id}>
                                User {id}
                            </option>
                        ))}
                    </select>
                </div>

                {userDetail && (
                    <div className="user-detail-card">
                        <h4>{userDetail.name}</h4>
                        <p><strong>Email:</strong> {userDetail.email}</p>
                        <p><strong>Phone:</strong> {userDetail.phone}</p>
                        <p><strong>Address:</strong> {userDetail.address}</p>
                    </div>
                )}
            </div>
        </div>
    );
};

export default DataFetchingDemo;
```

Langkah 4: Komponen Effect Dependencies Demo

File: src/components/DataFetchingDemo.jsx

EffectDependenciesDemo.jsx

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

const EffectDependenciesDemo = () => {
    const [count, setCount] = useState(0);
    const [name, setName] = useState('');
    const [windowWidth, setWindowWidth] = useState(window.innerWidth);
    const [timer, setTimer] = useState(0);
    const [logs, setLogs] = useState([]);

    const addLog = (message) => {
        setLogs(prev => [...prev, `${new Date().toLocaleTimeString()}: ${message}`]);
    };

    // Effect 1: Tanpa dependencies - run setelah setiap render
    useEffect(() => {
        addLog('Effect tanpa dependencies dijalankan');
    });

    // Effect 2: Empty dependencies - run sekali setelah mount
    useEffect(() => {
        addLog('Effect dengan [] dijalankan (mount)');

        return () => {
            addLog('Cleanup effect [] (unmount)');
        };
    }, []);

    // Effect 3: Dependency count - run ketika count berubah
    useEffect(() => {
        addLog(`Effect [count] dijalankan, count: ${count}`);

        return () => {
            addLog(`Cleanup effect [count] sebelumnya`);
        };
    }, [count]);

    // Effect 4: Dependency name - run ketika name berubah
    useEffect(() => {
        if (name) {
            addLog(`Effect [name] dijalankan, name: "${name}"`);
        }
    }, [name]);

    // Effect 5: Multiple dependencies - run ketika count atau name berubah
    useEffect(() => {
        addLog(`Effect [count, name] dijalankan, count: ${count}, name: "${name}"`);
    }, [count, name]);
```

```
48
49      // Effect 6: Window resize event listener dengan cleanup
50      useEffect(() => {
51          const handleResize = () => {
52              setWindowWidth(window.innerWidth);
53          };
54
55          addLog('Event listener resize dipasang');
56          window.addEventListener('resize', handleResize);
57
58          return () => {
59              addLog('Event listener resize dibersihkan');
60              window.removeEventListener('resize', handleResize);
61          };
62      }, []);
63
64      // Effect 7: Timer dengan cleanup
65      useEffect(() => {
66          addLog('Timer dimulai');
67
68          const interval = setInterval(() => {
69              setTimer(prev => prev + 1);
70          }, 1000);
71
72          return () => {
73              addLog('Timer dihentikan');
74              clearInterval(interval);
75          };
76      }, []);
77
78      const resetLogs = () => {
79          setLogs([]);
80      };
81
82      return (
83          <div className="dependencies-demo">
84              <h2>Demo Effect Dependencies</h2>
85
86              <div className="controls">
87                  <div className="control-group">
88                      <label>Count: {count}</label>
89                      <button onClick={() => setCount(prev => prev + 1)}
    className="btn btn-primary">
90                          +1 Count
91                      </button>
92                  </div>
93
94                  <div className="control-group">
95                      <label>Name:</label>
96                      <input
97                          type="text"
98                          value={name}
99                          onChange={(e) => setName(e.target.value)}
100                         placeholder="Ketik nama..."
```

```
                        className="text-input"
                    />
                </div>

                <div className="control-group">
                    <label>Window Width: {windowWidth}px</label>
                </div>

                <div className="control-group">
                    <label>Timer: {timer}s</label>
                </div>

                <button onClick={resetLogs} className="btn btn-secondary">
                    Reset Logs
                </button>
            </div>

            <div className="explanation">
                <h3>Penjelasan Dependencies:</h3>
                <ul>
                    <li><strong>No dependencies</strong> - Run setelah setiap render</li>
                    <li><strong>[]</strong> - Run sekali setelah mount</li>
                    <li><strong>[count]</strong> - Run ketika count berubah</li>
                    <li><strong>[name]</strong> - Run ketika name berubah</li>
                    <li><strong>[count, name]</strong> - Run ketika count ATAU name berubah</li>
                </ul>
            </div>

            <div className="logs-container">
                <h3>Effect Execution Logs:</h3>
                <div className="logs">
                    {logs.slice(-10).map((log, index) => (
                        <div key={index} className="log-entry">
                            {log}
                        </div>
                    ))}
                    {logs.length === 0 && <p>Belum ada logs...</p>}
                </div>
                <p className="log-count">Total logs: {logs.length}</p>
            </div>
        </div>
    );
};

export default EffectDependenciesDemo;
```

Langkah 5: Styling CSS

File: src/App.css

EffectDependenciesDemo.jsx

```css
/* File: src/App.css */
.App {
  max-width: 1200px;
  margin: 0 auto;
  padding: 20px;
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

/* Lifecycle Demo Styles */
.lifecycle-demo {
  background: #f8f9fa;
  padding: 20px;
  border-radius: 10px;
  margin-bottom: 30px;
  border-left: 4px solid #007bff;
}

.counter-section,
.visibility-section {
  margin: 20px 0;
  padding: 15px;
  background: white;
  border-radius: 8px;
  border: 1px solid #dee2e6;
}

.message-box {
  background: #e7f3ff;
  padding: 15px;
  border-radius: 5px;
  margin-top: 10px;
  border-left: 4px solid #007bff;
}

.explanation {
  background: #fff3cd;
  padding: 15px;
  border-radius: 5px;
  border: 1px solid #ffeaa7;
}

/* Data Fetching Demo Styles */
.data-fetching-demo {
  background: #f8f9fa;
  padding: 20px;
  border-radius: 10px;
  margin-bottom: 30px;
  border-left: 4px solid #28a745;
}
```

```css

.loading {
  display: flex;
  align-items: center;
  gap: 10px;
  background: #e7f3ff;
  padding: 15px;
  border-radius: 5px;
  margin: 10px 0;
}

.spinner {
  width: 20px;
  height: 20px;
  border: 2px solid #007bff;
  border-top: 2px solid transparent;
  border-radius: 50%;
  animation: spin 1s linear infinite;
}

@keyframes spin {
  0% {
    transform: rotate(0deg);
  }

  100% {
    transform: rotate(360deg);
  }
}

.error-message {
  background: #f8d7da;
  color: #721c24;
  padding: 15px;
  border-radius: 5px;
  margin: 10px 0;
  border: 1px solid #f5c6cb;
}

.users-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
  gap: 10px;
  margin: 15px 0;
}

.user-card {
  background: white;
  padding: 15px;
  border-radius: 8px;
  border: 2px solid #dee2e6;
  cursor: pointer;
  transition: all 0.3s ease;
}
```

```css

.user-card:hover {
  border-color: #007bff;
  transform: translateY(-2px);
}

.user-card.active {
  border-color: #28a745;
  background: #f8fff9;
}

.user-detail-section {
  margin-top: 20px;
  padding: 20px;
  background: white;
  border-radius: 8px;
  border: 1px solid #dee2e6;
}

.user-selector {
  margin-bottom: 15px;
}

.user-detail-card {
  background: #f8f9fa;
  padding: 15px;
  border-radius: 5px;
  border-left: 4px solid #28a745;
}

/* Dependencies Demo Styles */
.dependencies-demo {
  background: #f8f9fa;
  padding: 20px;
  border-radius: 10px;
  border-left: 4px solid #ffc107;
}

.controls {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
  gap: 15px;
  margin: 20px 0;
  padding: 20px;
  background: white;
  border-radius: 8px;
  border: 1px solid #dee2e6;
}

.control-group {
  display: flex;
  flex-direction: column;
  gap: 5px;
}
```

```css
.text-input {
  padding: 8px 12px;
  border: 2px solid #e9ecef;
  border-radius: 5px;
  font-size: 14px;
}

.text-input:focus {
  outline: none;
  border-color: #007bff;
}

.logs-container {
  margin-top: 20px;
}

.logs {
  max-height: 300px;
  overflow-y: auto;
  background: #2d3748;
  color: #e2e8f0;
  padding: 15px;
  border-radius: 5px;
  font-family: 'Courier New', monospace;
  font-size: 14px;
}

.log-entry {
  padding: 5px 0;
  border-bottom: 1px solid #4a5568;
}

.log-entry:last-child {
  border-bottom: none;
}

.log-count {
  text-align: center;
  margin-top: 10px;
  font-style: italic;
  color: #6c757d;
}

/* Common Button Styles */
.btn {
  padding: 10px 15px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  font-weight: bold;
  transition: all 0.3s ease;
}
```

```css
.btn-primary {
  background: #007bff;
  color: white;
}

.btn-secondary {
  background: #6c757d;
  color: white;
}

.btn-success {
  background: #28a745;
  color: white;
}

.btn:hover {
  opacity: 0.9;
  transform: translateY(-1px);
}

/* Responsive Design */
@media (max-width: 768px) {
  .controls {
    grid-template-columns: 1fr;
  }

  .users-grid {
    grid-template-columns: 1fr;
  }
}
```

Langkah 5: App.jsx

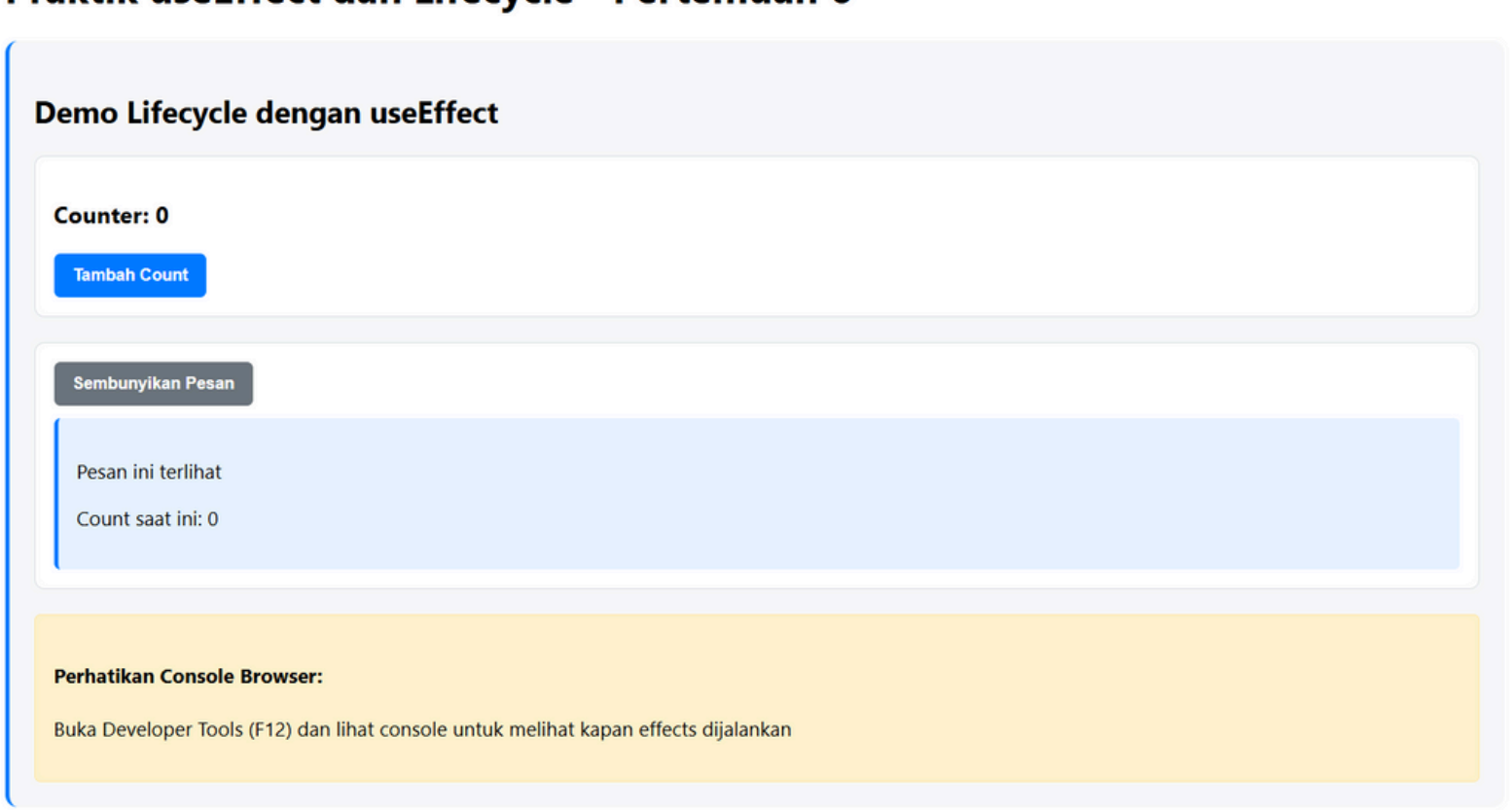
App.jsx

```jsx
import React from 'react';
import './App.css';
import LifecycleDemo from './components/LifecycleDemo';
import DataFetchingDemo from './components/DataFetchingDemo';
import EffectDependenciesDemo from './components/EffectDependenciesDemo';

function App() {
  return (
    <div className="App">
      <h1>Praktik useEffect dan Lifecycle - Pertemuan 6</h1>
      <LifecycleDemo />
      <DataFetchingDemo />
      <EffectDependenciesDemo />
    </div>
  );
}

export default App;
```

Tampilan



Demo Data Fetching dengan useEffect

Daftar Users

John Doe
john@example.com

Jane Smith
jane@example.com

Bob Johnson
bob@example.com

Alice Brown
alice@example.com

Detail User

Pilih User ID: User 1

User 1

Email: user1@example.com

Phone: +62 812-3456-7891

Address: Jl. Contoh No. 1, Jakarta

Demo Effect Dependencies

Count: 0

+1 Count

Name:

Ketik nama...

Window Width: 1600px

Timer: 39s

Reset Logs

Penjelasan Dependencies:

- **No dependencies** - Run setelah setiap render
- **[]** - Run sekali setelah mount
- **[count]** - Run ketika count berubah
- **[name]** - Run ketika name berubah
- **[count, name]** - Run ketika count ATAU name berubah

Effect Execution Logs:

```
5:09:42 PM: Effect tanpa dependencies dijalankan
5:09:42 PM: Effect tanpa dependencies dijalankan
5:09:42 PM: Effect tanpa dependencies dijalankan
5:09:42 PM: Effect tanpa dependencies dijalankan
5:09:42 PM: Effect tanpa dependencies dijalankan
5:09:42 PM: Effect tanpa dependencies dijalankan
5:09:42 PM: Effect tanpa dependencies dijalankan
5:09:42 PM: Effect tanpa dependencies dijalankan
5:09:42 PM: Effect tanpa dependencies dijalankan
5:09:42 PM: Effect tanpa dependencies dijalankan
```

Total logs: 51474