**Nama : I Komang Basudewa Suputra  
NIM : 42030071  
Mata Kuliah : Arsitektur Mikroservis**

**1. Skenario Unit Testing Untuk Fitur Pencarian Produk Berdasarkan Nama, Kategori, Dan Harga**  
 - Mencari Produk Berdasarkan Nama  
 - Mencari Produk Berdasarkan Kategori  
 - Mencari Produk Dengan Berdasarkan Harga Maksimum

**Produk database**

const products = [

{ id: 1, name: 'TV', category: 'Electronics', price: 1000 },

{ id: 2, name: 'Laptop', category: 'Electronics', price: 700 },

{ id: 3, name: 'Hat', category: 'Clothing', price: 15 },

{ id: 4, name: 'Jacket', category: 'Clothing', price: 50 },

{ id: 5, name: 'Book', category: 'Books', price: 10 },

];

**Fungsi untuk mencari produk**

function searchProducts(name, category, maxPrice) {

return products.filter((product) => {

return (

(!name || product.name.toLowerCase().includes(name.toLowerCase())) &&

(!category || product.category.toLowerCase() === category.toLowerCase()) &&

(!maxPrice || product.price <= maxPrice)

);

});

}

module.exports = searchProducts;

**Unit Testing Fitur Pencarian Produk**

const searchProducts = require('./searchProducts');

describe('searchProducts', () => {

test('Mencari produk berdasarkan nama', () => {

const result = searchProducts('TV');

expect(result).toEqual([

{ id: 1, name: 'TV', category: 'Electronics', price: 1000 },

]);

});

test('Mencari produk berdasarkan kategori', () => {

const result = searchProducts('', 'Electronics');

expect(result).toEqual([

{ id: 1, name: 'TV', category: 'Electronics', price: 1000 },

{ id: 2, name: 'Laptop', category: 'Electronics', price: 700 },

]);

});

test('Mencari produk berdasarkan harga maksimum', () => {

const result = searchProducts('', '', 20);

expect(result).toEqual([

{ id: 3, name: 'Hat', category: 'Clothing', price: 15 },

{ id: 5, name: 'Book', category: 'Books', price: 10 },

]);

});

**2. Component Testing untuk Fitur Pembayaran**

Langkah - langkah:

**- Memahami Komponen Payment System**

Komponen yang akan diuji adalah PaymentSystem. Ini adalah kelas yang digunakan untuk mengelola saldo dan berbagai transaksi pembayaran.

**- Persiapkan Alat Pengujian**

Dalam kode pengujian, saya menggunakan Jest sebagai alat pengujian untuk menjalankan pengujian komponen.

**- Pahami Fungsi Komponen**

PaymentSystem memiliki tiga fungsi utama: deposit (menyimpan dana ke saldo), makePayment (melakukan pembayaran), dan checkBalance (memeriksa saldo). Perilaku komponen ini telah dijelaskan dalam kode dan komentar.

**- Buat Berkas Pengujian PaymentSystem**

Dalam kode pengujian, saya telah membuat berkas terpisah dengan nama PaymentSystem.test.js untuk menguji komponen PaymentSystem.

**- Impor Komponen**

Komponen PaymentSystem diimpor dari berkas komponen yang sesuai di berkas pengujian.

**- Tentukan Skenario Pengujian**

Dalam kode pengujian, saya telah menentukan beberapa skenario pengujian, seperti deposit, pembayaran, dan pemeriksaan saldo. Skenario-skenario ini mencakup berbagai situasi yang mungkin terjadi.

**- Buat Pengujian**

Untuk setiap skenario pengujian, saya telah membuat pengujian yang menggunakan metode Jest test untuk menguji perilaku komponen dalam situasi tersebut. Pengujian menggunakan asersi (assertions) untuk memeriksa hasil.

**- Jalankan Pengujian**

Dalam kode pengujian, saya menggunakan Node JS untuk menjalankan pengujian. Pengujian dijalankan dengan menjalankan perintah npm test di terminal.

**- Uji Regresi**

Sebelumnya, saya telah menjalankan pengujian ulang setelah membuat perubahan pada komponen PaymentSystem untuk memastikan bahwa perubahan tersebut tidak memengaruhi perilaku yang sudah ada sebelumnya.

**- Analisis Hasil**

Setelah menjalankan pengujian, hasil pengujian akan dianalisis. Jika ada pengujian yang gagal, perlu dilakukan perbaikan pada komponen untuk memastikan bahwa perilaku sesuai dengan harapan.

**- Dokumentasi**

Selama pengujian, hasil pengujian dan tindakan perbaikan (jika diperlukan) didokumentasikan untuk referensi dan pemantauan selanjutnya.

**3. End-to-End Testing Alur Transaksi Search Produk, Add to Cart, dan Payment**

Prosedur End-to-End Testing untuk Alur Transaksi :

**- Perencanaan Pengujian**

Identifikasi alur transaksi yang akan diuji. Dalam kasus ini, alur transaksi melibatkan pencarian produk, menambahkannya ke keranjang, dan pembayaran.

Tentukan pengujian kasus uji (test cases) untuk setiap langkah dalam alur.

**- Persiapan Lingkungan Pengembangan**

Pastikan bahwa lingkungan pengujian (misalnya, server pengujian, database pengujian) sudah siap.

Pastikan bahwa kita memiliki akses ke aplikasi atau sistem yang akan diuji.

**- Inisialisasi Aplikasi**

Memulai dengan keadaan awal aplikasi yang bersih. Ini mungkin termasuk membersihkan keranjang belanja, menghapus data produk uji, dan mengatur saldo akun uji (jika relevan).

**- Eksekusi Pencarian Produk**

Buka aplikasi dan lakukan pencarian produk yang sesuai dengan skenario pengujian. Pastikan hasil pencarian sesuai dengan yang diharapkan.

**- Tambahkan Produk ke Keranjang**

Setelah menemukan produk yang sesuai, tambahkan produk tersebut ke keranjang belanja. Pastikan produk ditambahkan dengan benar.

**- Verifikasi Isi Keranjang**

a isi keranjang belanja untuk memastikan produk yang ditambahkan sesuai.

**- Lanjutkan ke Pembayaran**

Lanjutkan ke proses pembayaran dengan mengklik tombol "Bayar" atau langkah selanjutnya dalam aplikasi.

**- Isi Detail Pembayaran**

pembayaran dengan data uji yang sesuai (misalnya, nomor kartu kredit uji).

**- Verifikasi Pembayaran**

Setelah mengirim pembayaran, verifikasi bahwa "pembayaran berhasil diproses dan terima kasih atas pembayaran Anda".

**- Evaluasi Hasil**

Evaluasi hasil dari setiap langkah dalam alur transaksi. Pastikan bahwa setiap langkah berfungsi dengan benar sesuai dengan harapan.

**- Dokumentasi Hasil**

kumentasikan hasil dari pengujian, termasuk detail setiap langkah dan masalah yang ditemukan.

**- Uji Skenario Alternatif**

Selain skenario positif, uji skenario alternatif seperti pembayaran yang gagal, pembatalan transaksi, atau penanganan kesalahan lainnya.

**- Uji Kinerja**

Jika memungkinkan, uji kinerja alur transaksi dengan banyak pengguna secara bersamaan untuk melihat bagaimana aplikasi menangani beban.

**- Perbaikan dan Retesting**

Jika ditemukan masalah atau kesalahan, laporkan dan perbaiki masalah tersebut. Setelah perbaikan, lakukan pengujian ulang untuk memastikan bahwa masalah telah diperbaiki dengan benar.

**- Laporan Hasil**

Buat laporan hasil pengujian yang mencakup detail hasil pengujian, masalah yang ditemukan, dan langkah-langkah perbaikan yang telah diambil.