

Tugas I IF4052 Komputasi Layanan
Perancangan Model BPMN pada Aplikasi E-Commerce



Disusun oleh:
Kelompok 11

Yanuar Sano Nur Rasyid	13521110
Muhammad Habibi Husni	13521169
Dewana Gustavus Haraka Otang	13521173

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2024

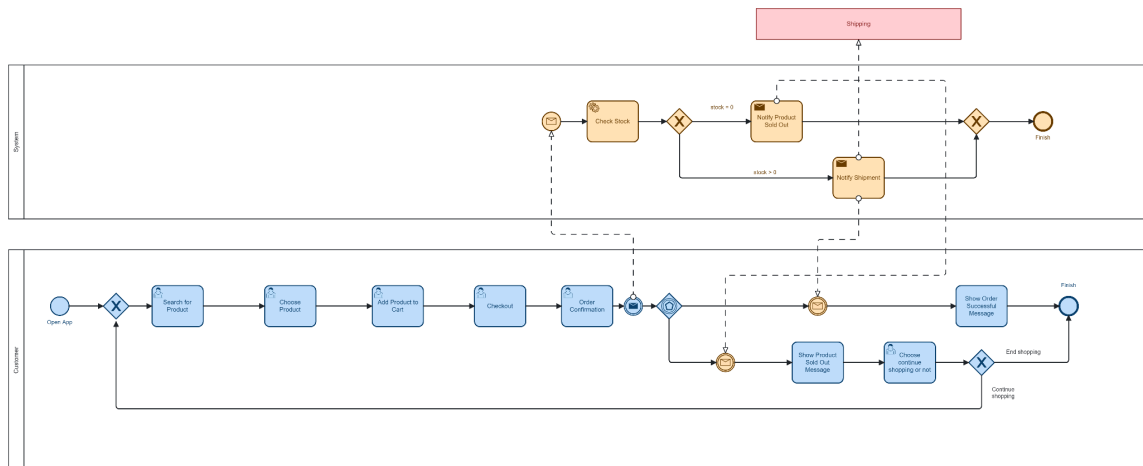
DAFTAR ISI

1. Latar Belakang	3
2. Model BPMN	3
3. Implementasi	4
4. Hasil Deployment	6
5. Lampiran	11

1. Latar Belakang

Aplikasi E-Commerce muncul sebagai bagian dari digitalisasi teknologi yang telah dapat mengubah berbagai aspek kehidupan, salah satunya adalah cara orang berbelanja. Di era yang hampir semua orang memiliki akses ke internet dan perangkat digital, jual beli offline semakin tergeser oleh kemudahan belanja online. Digitalisasi memungkinkan konsumen untuk berbelanja kapan saja dan di mana saja, tanpa terikat oleh lokasi atau waktu operasional toko fisik. Selain itu, perkembangan teknologi pembayaran dan logistik mendukung transaksi yang cepat, aman, dan efisien sehingga membuat interaksi jual beli offline menjadi semakin jarang dilakukan.

2. Model BPMN



Gambar 2.1. Model BPMN *E-Commerce*

Model BPMN *E-Commerce* yang dapat dilihat pada Gambar 2.1. memperlihatkan tiga aktor yang terlibat dalam proses bisnis, yaitu Customer, System dan Shipping. Customer atau pelanggan memulai aktivitas dengan membuka aplikasi *E-Commerce* dan mencari produk di dalam aplikasi dengan memasukkan nama produk yang ingin dicari ke dalam Search Form. Setelah itu, aplikasi akan menampilkan daftar produk yang dicari dan Customer dapat memilih suatu produk untuk melihat detailnya lebih lanjut di dalam Choose Product Form. Detail produk akan ditampilkan, seperti gambar, oleh aplikasi yang kemudian dapat dimasukkan ke keranjang oleh Customer dengan mengisi Add To Cart Form. Produk-produk yang sudah dimasukkan oleh Customer dapat di-checkout oleh Customer dengan mengisi Checkout Form. Sebelum *order* diproses oleh System, aplikasi

akan meminta konfirmasi kepada Customer apakah pesanan produknya sudah sesuai dengan mengisi Confirmation Form.

Proses dilanjutkan dengan System yang akan memproses form yang sebelumnya dikirim oleh aplikasi Customer dengan mengecek apakah stok dari produk yang diorder tersedia atau tidak. Apabila stok produk yang dipesan tersedia, System akan melanjutkan proses dengan aktivitas menyiapkan produk dan mengirimkan pesan pengiriman kepada Shipping dan Customer untuk melanjutkan proses pembelian. Shipping akan menerima pesan untuk dilanjutkan ke proses pengiriman kepada Customer, sedangkan Customer akan menerima notifikasi bahwa order berhasil dipesan sebagai penanda bahwa proses bisnis sudah selesai dilakukan dan dapat menutup aplikasi. Apabila stok produk yang dipesan tidak tersedia, System akan mengirimkan pesan kepada Customer bahwa produknya sudah habis yang kemudian akan ditampilkan notifikasinya oleh aplikasi. Kemudian, Customer memiliki dua pilihan yaitu menyelesaikan penggunaan aplikasi dengan tidak memilih untuk melanjutkan pembelian yang berarti menyelesaikan proses bisnis atau kembali ke aktivitas mencari barang yang lain jika memilih untuk melanjutkan pembelian.

3. Implementasi

Dalam implementasi terdapat 4 buah *Web Services* yang akan memproses task-task penting dalam proses.

Web service `confirmOrder` bertugas menerima dan memproses *task* “Checkout” yang dilakukan oleh Customer. *Worker* akan meneruskan informasi product yang di-order Customer kepada System. Pada implementasi *web service*, *worker* akan menerima *argument* job yang nantinya dapat mengekstrak informasi produk pada variable `order`. *Worker* lalu akan memanggil fungsi `publish_message` untuk mengirim informasi kepada System.

```
@router.task(task_type="confirmOrder")
async def confirmOrder(job: Job, *args, **kwargs):
    print("confirmOrder Job")
```

```

product = job.variables.get("order")
print("product :", product)
await client.publish_message(
    name="item-order",
    correlation_key="item-order-"+product,
    variables={ "product" : product })
print("ok")
return

```

Web service checkStock bertugas menerima dan memproses *task* “Check Stock” yang dilakukan oleh System. *Worker* akan menerima informasi product yang di-checkout Customer yang nantinya akan dicek ketersediaan stoknya. Pada implementasi *web service*, *worker* akan mengambil nilai yang tersimpan pada variable stock. *Worker* lalu akan mengecek jumlah stok yang dimiliki barang yang akan dibeli dan mengembalikan jumlah stok sebagai nilai pada return function. Pada implementasi *Worker* kali ini kami menggunakan nilai *dummy* yaitu stok berjumlah 1 apabila produk yang dibeli merupakan meja dan stok habis apabila produk yang dibeli bukan meja.

```

@router.task(task_type="checkStock", single_value=True, variable_name="stock")
async def checkStock(job: Job, *args, **kwargs) -> int:
    print("checkStock Job")
    product : str = job.variables.get("product")
    print("product :", product)
    if (product.find("meja") != -1):
        print("current stock : 1")
        return 1
    else:
        print("current stock : 0")
        return 0

```

Web service notifySoldOut bertugas menerima dan memproses *task* “Notify Product Sold Out” yang dilakukan oleh System. *Worker* akan mengirim notifikasi kepada Customer bahwa produk yang ingin dibeli sudah kehabisan stok. Pada implementasi *web service*, *worker* akan menerima *argument* job yang nantinya dapat mengekstrak informasi nama produk pada variable product. *Worker* lalu akan memanggil fungsi `publish_message` untuk mengirim informasi kepada Customer bahwa produk yang ingin dibeli sudah habis.

```
@router.task(task_type="notifySoldOut")
async def notifySoldOut(job: Job, *args, **kwargs):
    print("notifySoldOut Job")
    product = job.variables.get("product")
    print("product :", product)
    await client.publish_message(
        name="sold_out",
        correlation_key=product)
    print("ok")
    return
```

Web service notifySuccessful bertugas menerima dan memproses *task* “Notify Shipment” yang dilakukan oleh Service. *Worker* akan mengirim notifikasi kepada Customer bahwa *order*nya berhasil diproses dan juga mengirim notifikasi kepada Shipping untuk memproses pengiriman barang kepada Customer. Pada implementasi *web service*, *worker* akan menerima *argument* job yang nantinya dapat mengekstrak informasi produk pada variable product. *Worker* lalu akan memanggil fungsi publish_message untuk mengirim informasi kepada Customer dan Shipping.

```
@router.task(task_type="notifySuccessful")
async def notifySuccessful(job: Job, *args, **kwargs):
    print("notifySuccessful Job")
    product = job.variables.get("product")
    print("product :", product)
    await client.publish_message(
        name="successful",
        correlation_key=product)
    print("ok")
    return
```

4. Hasil Deployment

Berikut merupakan hasil *deployment* BPMN:

Search for Product

Fix Diagram

Assigned to me

Unassign

Task

Process

Product Name

meja

Complete Task

Gambar 4.1 Tampilan Form Mencari Barang

Choose Product

Fix Diagram

Assigned to me

Unassign

Task

Process

Search Result :

☒ meja kayu

☐ meja besi

Complete Task


Gambar 4.2 Tampilan Form Memilih Barang

Add Product to Cart
Fix Diagram

Assigned to meUnassign

TaskProcess

meja kayu



Complete Task

Gambar 4.3 Tampilan Form Memasukkan Barang ke Keranjang

Checkout

Fix Diagram

Assigned to me

Unassign

Task

Process

Your Cart

– meja kayu

Complete Task

Gambar 4.4 Tampilan Form Checkout

Order Confirmation

Fix Diagram

Assigned to me

Unassign

Task

Process

Confirm Your Order

– meja kayu

Complete Task

Gambar 4.5 Tampilan Form Konfirmasi Pesanan

Choose continue shopping or not

Fix Diagram

Assigned to me

Unassign

Task

Process

Product Sold Out

Continue Shopping?*

☒

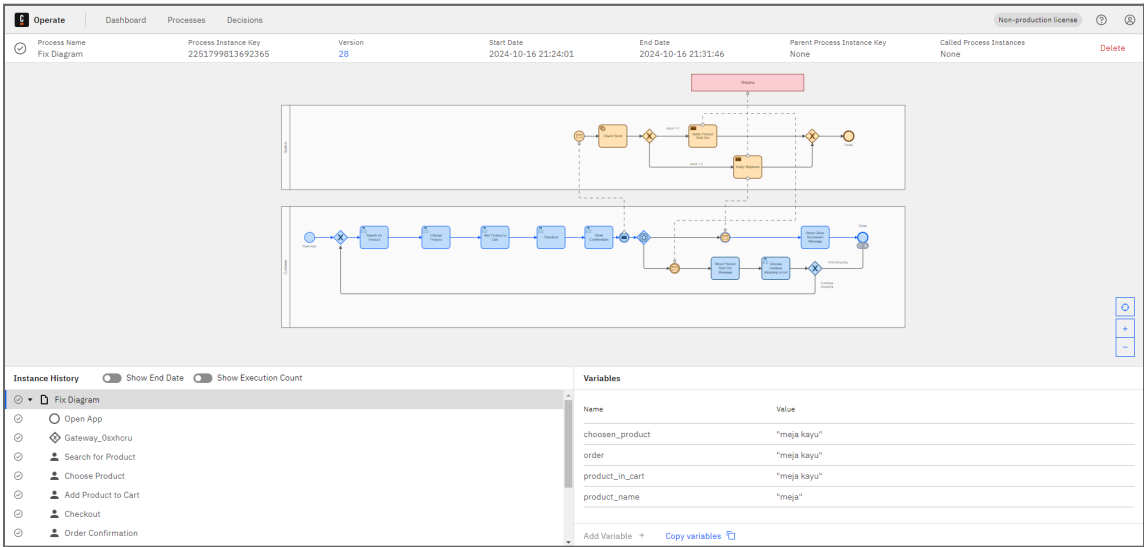
Continue

☐

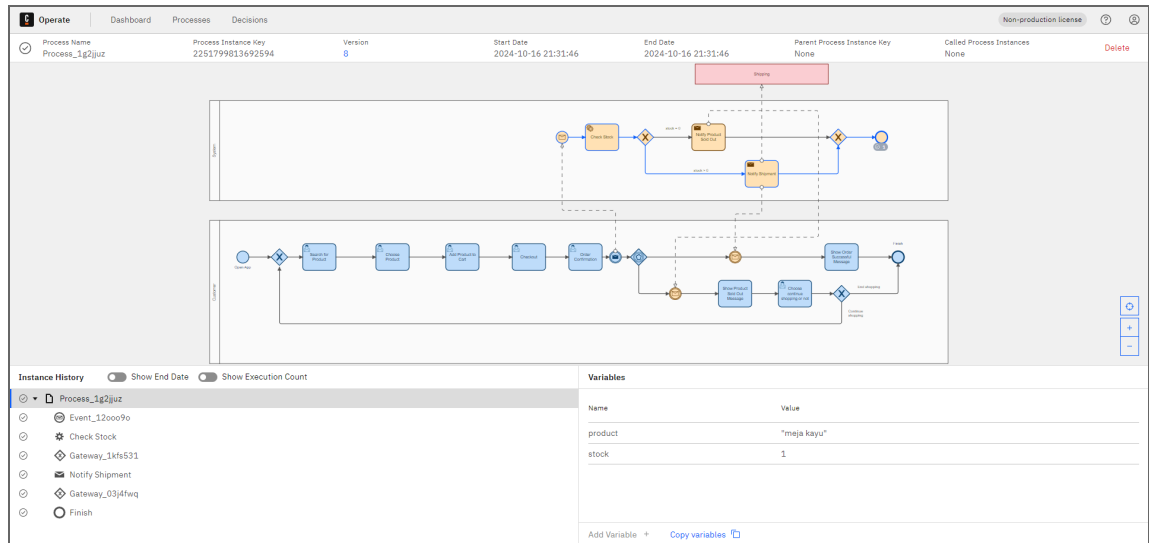
Stop

Complete Task

Gambar 4.6 Tampilan Form Memilih Melanjutkan Belanja



Gambar 4.7 Hasil Eksekusi Proses *Customer*



Gambar 4.8 Hasil Eksekusi Proses System

5. Lampiran

[yansans/BPMN-E-Commerce: Tugas I IF4052 Komputasi Layanan Perancangan Model BPMN pada aplikasi E-Commerce \(github.com\)](#)