# Laporan Week 11: Pengujian White-Box Marketplace UMKM Lokal (Kelompok D)

Efraim Melvyn Rafelino Lahai - 5803024009

Kode dan test dalam laporan ini adalah versi sederhana dari logika Order dan Catalog pada SRS Marketplace UMKM Lokal v0.6, ditulis dalam Python dan diuji dengan PyTest.

| Bagian | Berkas |
|---|---|
| Modul prosedural | `umkm/checkout.py`, fungsi `validate_checkout` |
| Class dengan state | `umkm/order.py`, class `Order` |
| Hierarki pewarisan | `umkm/catalog.py`, class `Product`, `FoodProduct`, `ClothingProduct` |
| Test | `tests/test_checkout_paths.py`, `tests/test_order_state.py`, `tests/test_product_polymorphism.py` |

Perintah untuk menjalankan test dan coverage:

```
python -m pytest --cov=umkm --cov-branch --cov-report=term-missing --cov-report=html:htmlcov
```

## 1. Control flow graph, V(G), dan jalur independen

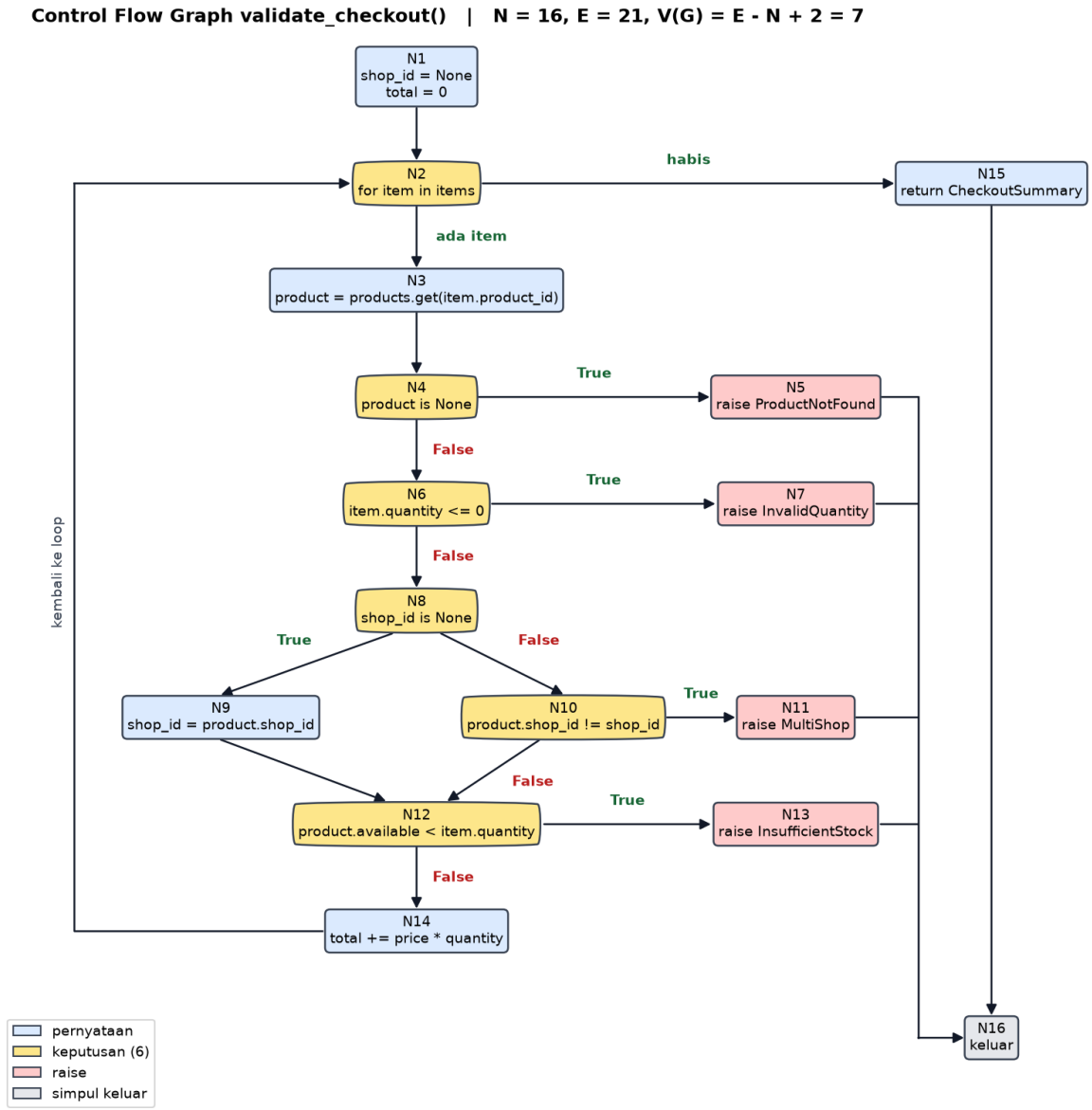
### 1.1 Fungsi yang dianalisis

`validate_checkout` memeriksa keranjang terhadap katalog, lalu menghitung total harga (FR-ORD-02, FR-ORD-03, FR-ORD-08). Harga diambil dari katalog, stok harus cukup, dan semua item harus berasal dari satu toko. Fungsi ini memiliki 6 keputusan.

```
def validate_checkout(items, products):
    shop_id = None                                  # N1
    total = 0
    for item in items:                              # N2  keputusan 1
        product = products.get(item.product_id)     # N3
        if product is None:                         # N4  keputusan 2
            raise ProductNotFoundError(...)         # N5
        if item.quantity <= 0:                      # N6  keputusan 3
            raise InvalidQuantityError(...)         # N7
        if shop_id is None:                         # N8  keputusan 4
            shop_id = product.shop_id               # N9
        elif product.shop_id != shop_id:            # N10 keputusan 5
            raise MultiShopError(...)               # N11
        if product.available < item.quantity:       # N12 keputusan 6
            raise InsufficientStockError(...)       # N13
        total += product.price * item.quantity      # N14
    return CheckoutSummary(shop_id, total)          # N15
```

N16 adalah simpul keluar; semua `raise` dan `return` menuju ke sana.

## 1.2 Control flow graph




*Gambar 1. Control flow graph validate_checkout*

## 1.3 Perhitungan V(G)

Graf memiliki 16 simpul (N) dan 21 edge (E): 1→2, 2→3, 2→15, 3→4, 4→5, 4→6, 5→16, 6→7, 6→8, 7→16, 8→9, 8→10, 9→12, 10→11, 10→12, 11→16, 12→13, 12→14, 13→16, 14→2, 15→16.

- V(G) = E − N + 2 = 21 − 16 + 2 = **7**
- V(G) = jumlah simpul keputusan + 1 = 6 + 1 = **7**
- V(G) = jumlah region = 6 region tertutup + 1 region luar = **7**

## 1.4 Jalur independen

Ada 7 jalur independen. Setiap jalur melewati minimal satu edge yang belum dilewati jalur sebelumnya.

| Jalur | Simpul yang dilewati | Edge baru | Skenario |
|---|---|---|---|
| P1 | 1-2-15-16 | 2→15 | Keranjang kosong, total 0 |
| P2 | 1-2-3-4-5-16 | 4→5 | Produk tidak ada di katalog |
| P3 | 1-2-3-4-6-7-16 | 6→7 | Kuantitas 0 atau negatif |
| P4 | 1-2-3-4-6-8-9-12-13-16 | 12→13 | Stok tidak cukup pada item pertama |
| P5 | 1-2-3-4-6-8-9-12-14-2-15-16 | 12→14, 14→2 | Satu item valid |
| P6 | 1-2-3-4-6-8-9-12-14-2-3-4-6-8-10-12-14-2-15-16 | 8→10, 10→12 | Dua item dari toko yang sama |
| P7 | 1-2-3-4-6-8-9-12-14-2-3-4-6-8-10-11-16 | 10→11 | Dua item dari toko berbeda |

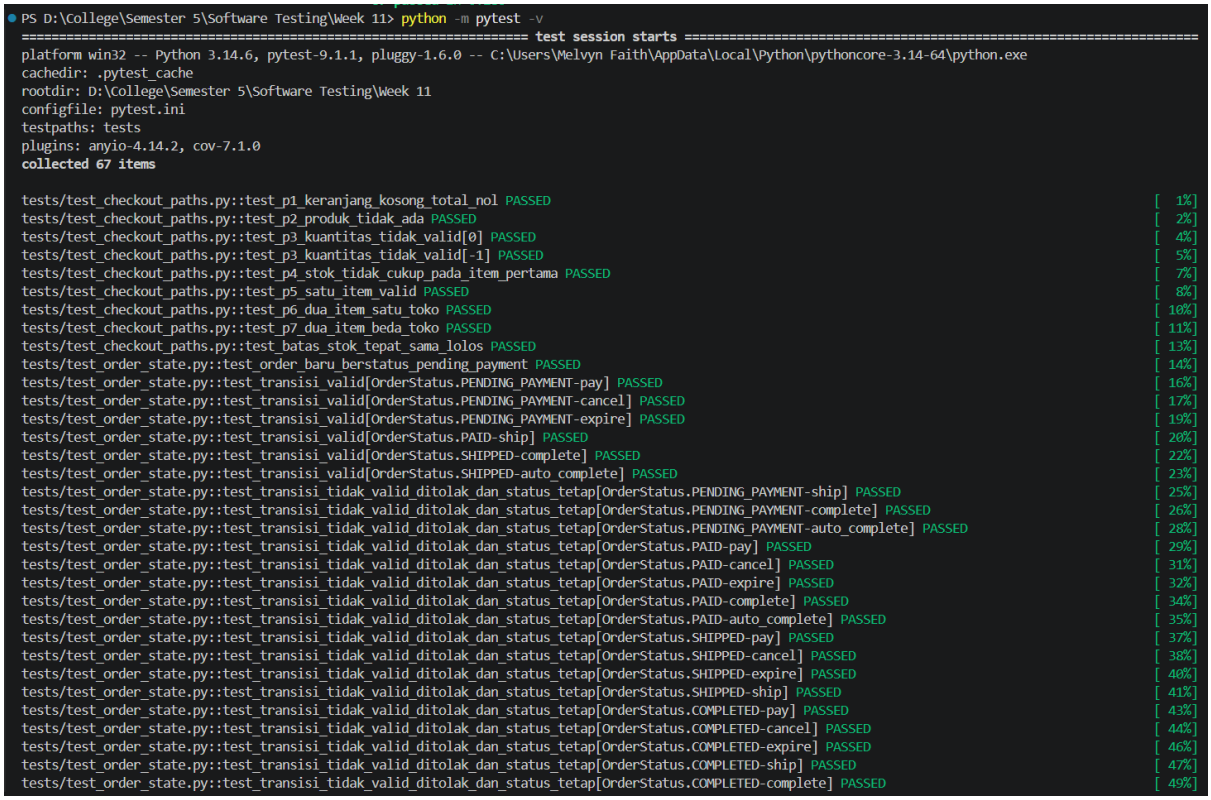
Pengecekan keranjang kosong sengaja tidak ditulis sebagai `if not items` sebelum loop. Dengan guard itu, jalur "loop tidak dimasuki" tidak mungkin dijalankan dan salah satu jalur basis menjadi mustahil. Keranjang kosong dibiarkan menghasilkan total 0, dan penolakannya menjadi tugas pemanggil.

## 2. Test PyTest untuk semua jalur

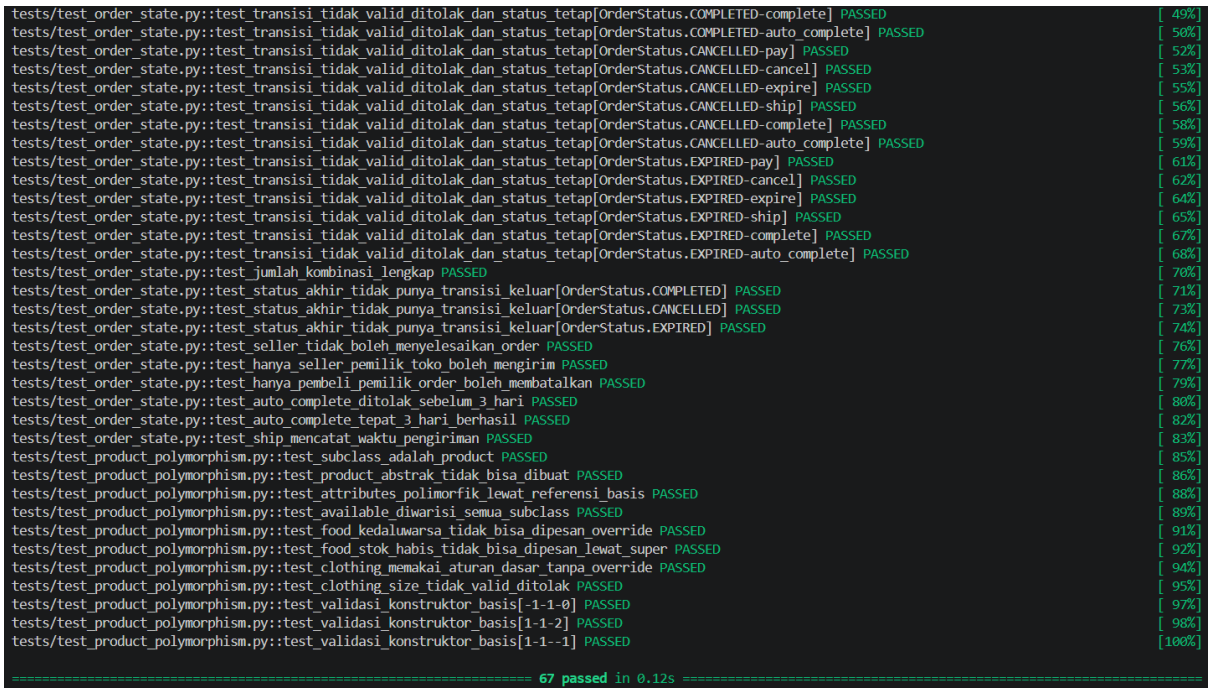
`tests/test_checkout_paths.py` berisi satu test untuk tiap jalur, ditambah satu test nilai batas (stok tepat sama dengan jumlah pesanan harus lolos). Jalur P3 diuji dengan dua nilai, 0 dan −1.

| Jalur | Test |
|---|---|
| P1 | `test_p1_keranjang_kosong_total_nol` |
| P2 | `test_p2_produk_tidak_ada` |
| P3 | `test_p3_kuantitas_tidak_valid` (0 dan −1) |
| P4 | `test_p4_stok_tidak_cukup_pada_item_pertama` |
| P5 | `test_p5_satu_item_valid` |
| P6 | `test_p6_dua_item_satu_toko` |
| P7 | `test_p7_dua_item_beda_toko` |
| Batas | `test_batas_stok_tepat_sama_lolos` |

Seluruh test lulus (67 test untuk ketiga berkas).



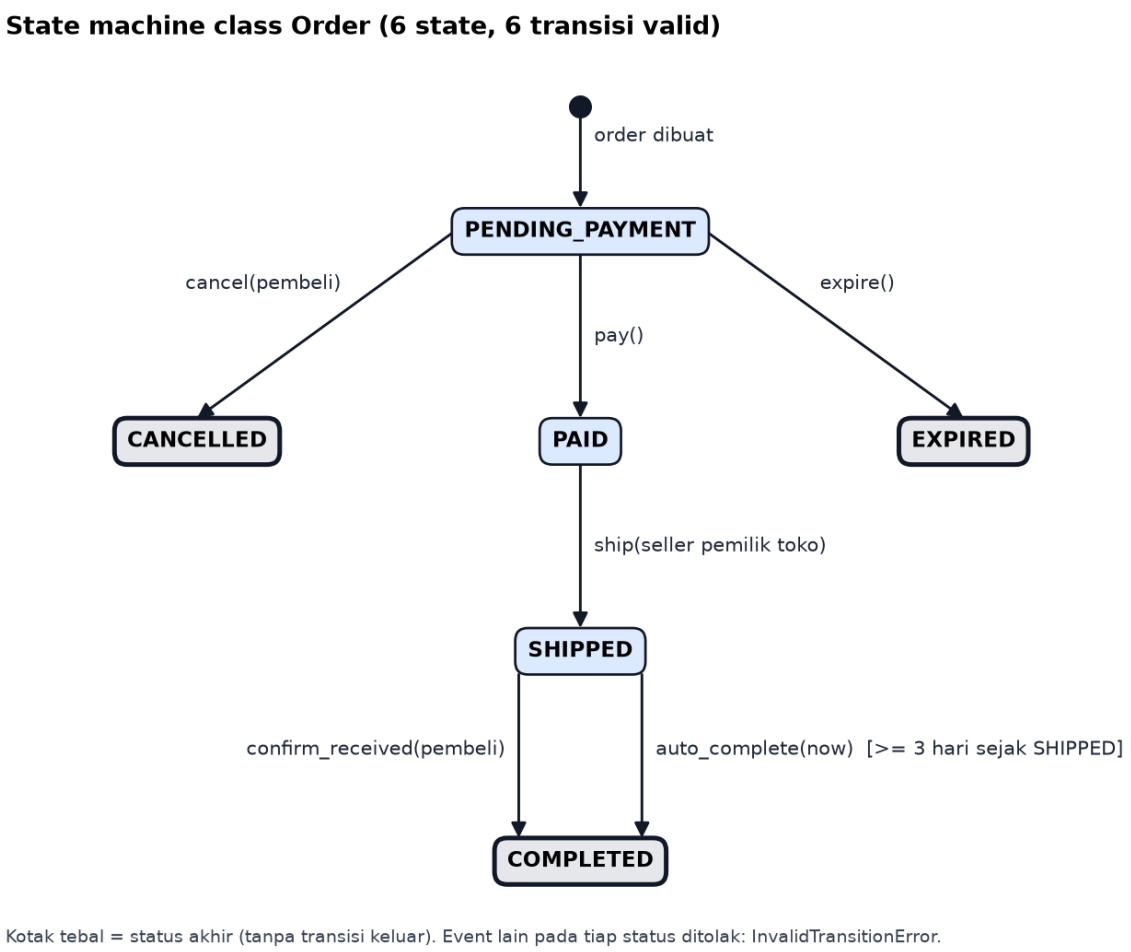

*Gambar 2. Hasil pytest -v, bagian 1*




*Gambar 3. Hasil pytest -v, bagian 2*

## 3. State machine dan tabel transisi

Class `Order` memiliki 6 state dan 6 transisi valid (FR-ORD-04, FR-ORD-09, FR-ORD-10). Lima transisi berasal dari SRS. Satu lagi, `auto_complete`, adalah penyelesaian otomatis 3 hari setelah `SHIPPED` (FR-ORD-10).




*Gambar 4. State machine Order*

Tabel transisi. Tanda ✗ berarti event ditolak dengan `InvalidTransitionError` dan status tidak berubah.

| Status \ Event | `pay` | `cancel` | `expire` | `ship` | `complete` | `auto_complete` |
|---|---|---|---|---|---|---|
| PENDING_PAYMENT | PAID | CANCELLED | EXPIRED | ✗ | ✗ | ✗ |
| PAID | ✗ | ✗ | ✗ | SHIPPED | ✗ | ✗ |
| SHIPPED | ✗ | ✗ | ✗ | ✗ | COMPLETED | COMPLETED (≥ 3 hari) |
| COMPLETED | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| CANCELLED | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| EXPIRED | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |

Dari 36 kombinasi, 6 valid dan 30 tidak valid. COMPLETED, CANCELLED, dan EXPIRED adalah status akhir.

Selain status, tiga event memeriksa pelaku dan menolak dengan `PermissionDeniedError`: `cancel` dan `complete` hanya untuk pembeli pemilik order (seller tidak boleh menyelesaikan order), `ship` hanya untuk seller pemilik toko. Pemeriksaan pelaku dilakukan sebelum pemeriksaan status.

## 4. Test class

`tests/test_order_state.py` (47 test):

- 6 transisi valid dan 30 transisi tidak valid, diuji dengan parametrisasi atas seluruh matriks 6 × 6. Tiap transisi tidak valid juga memeriksa bahwa status tidak berubah.
- Status akhir tidak punya transisi keluar.
- Pelaku yang tidak berhak ditolak (seller menyelesaikan order, seller lain mengirim, pembeli lain membatalkan).
- `auto_complete` ditolak sebelum 3 hari dan berhasil tepat pada 3 hari.

`tests/test_product_polymorphism.py` (11 test) menguji pewarisan dan polimorfisme pada hierarki `Product`. `Product` adalah kelas abstrak dengan method abstrak `attributes()` (AR-01, FR-CAT-03); `FoodProduct` dan `ClothingProduct` mewarisinya.

- Polimorfisme: `attributes()` dipanggil lewat referensi bertipe `Product` dan menghasilkan `expiryDate` untuk makanan dan `size` untuk pakaian.
- Override: `FoodProduct.is_orderable` memanggil `super()` lalu menambah syarat belum kedaluwarsa. Pada hari kedaluwarsa produk masih boleh dipesan, sehari setelahnya tidak.
- Pewarisan tanpa override: `ClothingProduct` memakai `is_orderable` dan `available` dari kelas dasar.
- `Product` tidak dapat dibuat langsung (`TypeError`), dan nilai tidak valid pada konstruktor ditolak (`ValueError`).

## 5. Laporan --cov-branch dan analisis

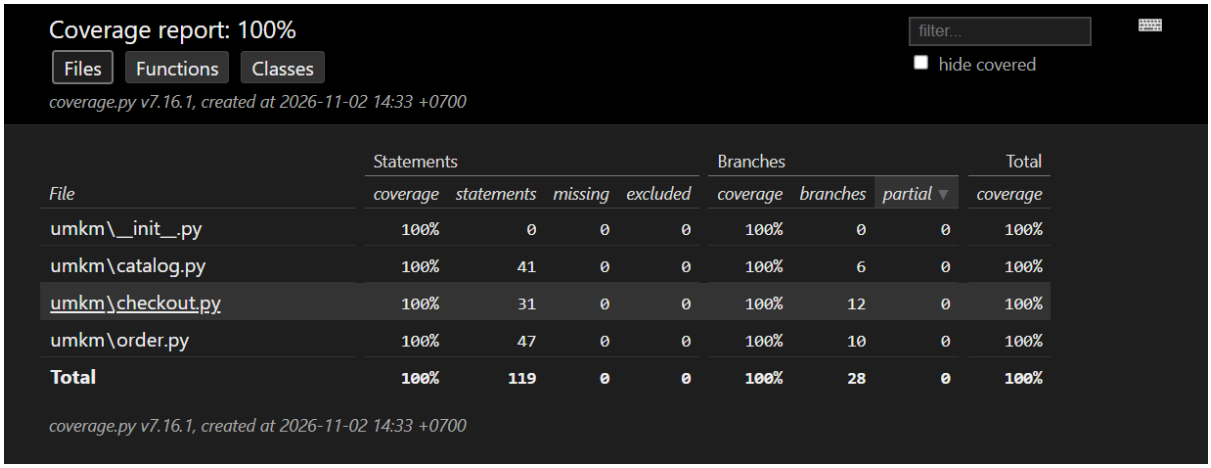
```
PS D:\College\Semester 5\Software Testing\Week 11> python -m pytest --cov=umkm --cov-branch --cov-report=term-missing --cov-report=html:htmlcov
============================================ test session starts ============================================
platform win32 -- Python 3.14.6, pytest-9.1.1, pluggy-1.6.0
rootdir: D:\College\Semester 5\Software Testing\Week 11
configfile: pytest.ini
testpaths: tests
plugins: anyio-4.14.2, cov-7.1.0
collected 67 items

tests\test_checkout_paths.py .........                                                          [ 13%]
tests\test_order_state.py ..............................................                        [ 83%]
tests\test_product_polymorphism.py ...........                                                  [100%]

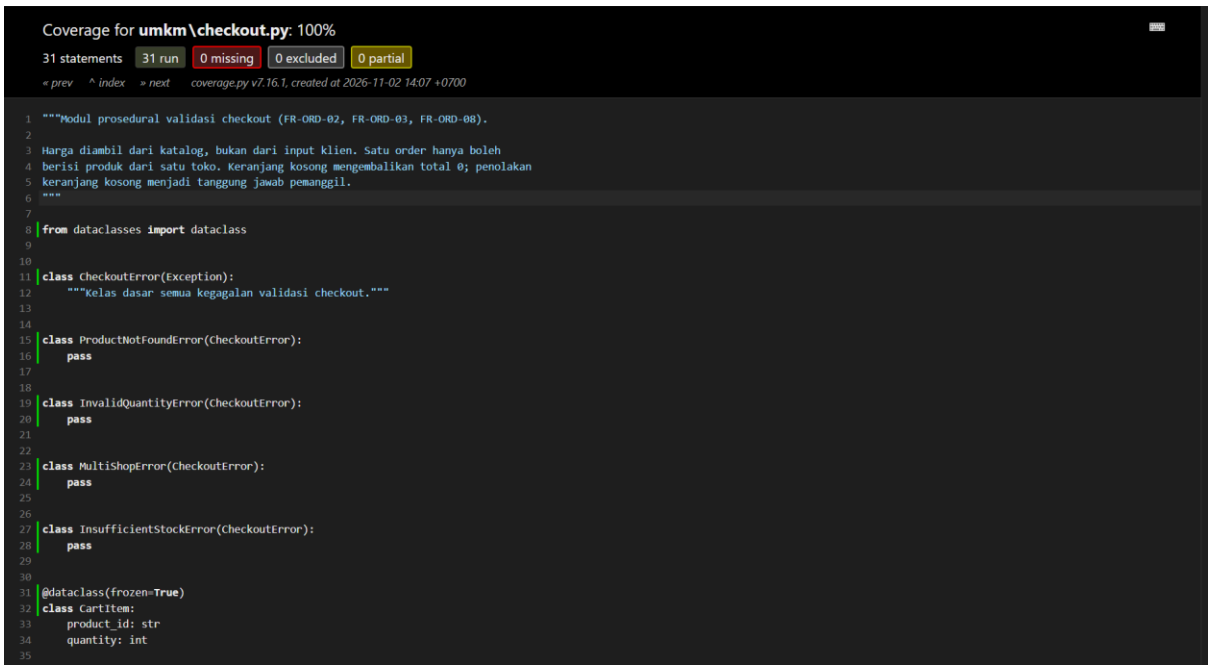
================================================ tests coverage =================================================
_____________________________ coverage: platform win32, python 3.14.6-final-0 _____________________________

Name              Stmts   Miss Branch BrPart  Cover   Missing
-------------------------------------------------------------
umkm\__init__.py      0      0      0      0   100%
umkm\catalog.py      41      0      6      0   100%
umkm\checkout.py     31      0     12      0   100%
umkm\order.py        47      0     10      0   100%
-------------------------------------------------------------
TOTAL               119      0     28      0   100%
Coverage HTML written to dir htmlcov
============================================ 67 passed in 0.28s =============================================
```

*Gambar 5. Hasil pytest dengan --cov-branch*




*Gambar 6. Laporan HTML coverage*




*Gambar 7. Detail coverage checkout.py*

Semua modul mencapai 100% statement dan 100% branch (28 dari 28 branch). Jumlah branch sesuai hitungan manual: `checkout.py` 12 (6 keputusan × 2 arah), `order.py` 10, dan `catalog.py` 6.

## Analisis

**Branch coverage tidak sama dengan path coverage.** Untuk memeriksanya, satu test dilepas dari `test_checkout_paths.py` lalu coverage dijalankan ulang:

- Tanpa P6 (dua item, toko sama), coverage `checkout.py` turun menjadi 98% dengan satu branch parsial `55->57`. Cabang `elif` bernilai False hanya tercapai pada iterasi kedua loop, jadi P6 memang dibutuhkan.
- Tanpa P1 (keranjang kosong), coverage tetap 100%. Edge `2→15` sudah dilewati di akhir loop pada test lain. Jalur P1 tetap perlu diuji karena perilakunya berbeda, tetapi branch coverage tidak bisa menunjukkan bahwa jalur itu hilang.

Jadi 100% branch coverage belum menjamin seluruh jalur teruji. Jaminan itu datang dari basis set hasil analisis V(G).

**Kondisi majemuk tidak terukur penuh.** `coverage.py` menghitung branch per baris, bukan per kondisi. Ekspresi `super().is_orderable(today) and today <= self.expiry_date` dan `self.status is S.SHIPPED and now - self.shipped_at < AUTO_COMPLETE_AFTER` tidak diukur sampai tingkat condition coverage atau MC/DC. Karena itu test untuk kondisi tersebut dibuat manual: makanan kedaluwarsa dengan stok ada, makanan segar dengan stok habis, `auto_complete` pada status selain SHIPPED, dan batas tepat 3 hari.

**Coverage tinggi belum tentu berarti test efektif.** Untuk menguji ini, 9 mutan dibuat secara manual, masing-masing dengan satu perubahan kecil, misalnya `<= 0` menjadi `< 0`, `<` menjadi `<=` pada stok dan pada batas 3 hari, `!=` menjadi `==` pada pengecekan toko, `price * quantity` menjadi `price + quantity`, 3 hari menjadi 2 hari, dan penambahan transisi `PAID → CANCELLED`. Kesembilannya membuat test gagal, sehingga test dapat mendeteksi perubahan tersebut. Jumlah mutan ini kecil dan dibuat tangan, jadi hasilnya bukan pengganti alat mutation testing seperti `mutmut`.