

UNIT TESTING VỚI AI PROMPT

BLOOD REQUEST



FPT UNIVERSITY

SE19B03

THÀNH VIÊN

- ① TÙ NGUYỄN ĐỨC ANH - DE190195
- ② NGÔ QUỐC TRUNG - DE190279
- ③ LÊ QUỐC ĐẠT - DE190277
- ④ NGUYỄN NGỌC THIỆN - DE190261
- ⑤ HOÀNG LÂN - DE190424

CÁC BƯỚC THỰC HIỆN

- ① PHÂN TÍCH & CHỌN FEATURE
- ② THIẾT KẾ TEST CASES
- ③ SINH TEST CODE (JEST) – MẪU THAM CHIẾU
- ④ DEBUG TESTS
- ⑤ MOCKING & OPTIMIZATION
- ⑥ DOCUMENTATION & DEMO

1) PHÂN TÍCH & CHỌN FEATURE

Bối cảnh dự án

- Tech stack: **Java (Maven), JUnit 5, Mockito, JaCoCo.**
- Module chính cần test: BloodRequestService.
- Các dependency đều là **interface** để mock:
 - HospitalRepository, BloodRequestRepository, AdminRepository,
 - BloodInventoryRepository
 - EmailService, NotificationService

1) PHÂN TÍCH & CHỌN FEATURE

5 hàm trọng tâm

1. createBloodRequest(hospitalId, bloodType, quantity)
2. approveRequest(requestId, adminId)
3. rejectRequest(requestId, reason)
4. fulfillRequest(requestId)
5. notifyHospital(requestId)

1) PHÂN TÍCH & CHỌN FEATURE

Quy tắc nghiệp vụ (suy ra từ interface/model và test đã có)

- **Trạng thái:** PENDING → APPROVED/REJECTED → (COMPLETED khi fulfill thành công).
- **create:** hospital phải tồn tại, bloodType không rỗng, quantity > 0, không có request đang PENDING cùng (hospitalId, bloodType).
- **approve:** admin hợp lệ, request tồn tại & đang PENDING → cập nhật APPROVED + set approvedBy.
- **reject:** request PENDING, có reason hợp lệ → cập nhật REJECTED + lưu reason.
- **fulfill:** request APPROVED, tồn kho đủ quantity → deduct kho, cập nhật COMPLETED.
- **notifyHospital:** chỉ thông báo khi trạng thái ∈ {APPROVED, REJECTED, COMPLETED}; cần email hợp lệ; thành công khi **cả gửi email & push notification đều true**.

2) Thiết kế Test Cases (Ma trận)

Function	ID	Case Name	Given	When	Then	Notes (mock/verify)
createBloodRequest	C1	create_hospital_OK_O+_qty3_true	<code>hospitals.existsById(h)=true , requests.existsPending(h,bt)=false , bloodType="O+" , qty=3</code>	create	<code>true ; saved PENDING</code>	<code>verify(requests).save(argThat(r -> r.getStatus()==PENDING && r.getQuantity()==3))</code>
createBloodRequest	C2	create_hospital_not_found_false	<code>hospitals.existsById(h)=false</code>	create	<code>false</code>	<code>verify(requests, never()).save(any())</code>
createBloodRequest	C3	create_qty_le_0_false	<code>qty=0</code>	create	<code>false</code>	<code>verify(requests, never()).save(any())</code>
createBloodRequest	C4	create_bloodType_invalid_false	<code>bloodType="x0"</code>	create	<code>false</code>	<code>verify(requests, never()).save(any())</code>
createBloodRequest	C5	create_duplicate_PENDING_false	<code>existsPending(h,bt)=true</code>	create	<code>false</code>	<code>verify(requests, never()).save(any())</code>
createBloodRequest	C6	create_quantity_boundary_min_true	<code>qty=1 , existsById=true , existsPending=false</code>	create	<code>true</code>	<code>verify(requests).save(argThat(r -> r.getQuantity()==1 && r.getStatus()==PENDING))</code>
approveRequest	A1	approve_pending_adminOK_true	<code>admins.existsById(admin)=true , requests.findById(id)=PENDING</code>	approve	<code>true ; status APPROVED</code>	<code>verify(requests).updateStatus(id, APPROVED, admin, null)</code>
approveRequest	A2	approve_not_found_false	<code>requests.findById(id)=empty</code>	approve	<code>false</code>	<code>verify(requests, never()).updateStatus(anyInt(), any(), any(), any())</code>
approveRequest	A3	approve_status_not_pending_false	<code>requests.findById(id).status=APPROVED</code>	approve	<code>false</code>	<code>verify(requests, never()).updateStatus(anyInt(), any(), any(), any())</code>
approveRequest	A4	approve_admin_invalid_false	<code>admins.existsById(admin)=false</code>	approve	<code>false</code>	<code>verifyNoInteractions(requests)</code>

2) Thiết kế Test Cases (Ma trận)

rejectRequest	R1	reject_pending_reasonOK_true	<code>requests.findById(id)=PENDING , reason="dup"</code>	reject	<code>true ; status REJECTED</code>	<code>verify(requests).updateStatus(id, REJECTED, null, "dup")</code>
rejectRequest	R2	reject_reason_blank_false	<code>reason=" "</code>	reject	<code>false</code>	<code>verify(requests, never()).updateStatus(anyInt(), any(), any(), any())</code>
rejectRequest	R3	reject_reason_null_false	<code>reason=null</code>	reject	<code>false</code>	<code>verify(requests, never()).updateStatus(anyInt(), any(), any(), any())</code>
rejectRequest	R4	reject_not_found_false	<code>requests.findById(id)=empty</code>	reject	<code>false</code>	<code>verify(requests, never()).updateStatus(anyInt(), any(), any(), any())</code>
rejectRequest	R5	reject_status_not_pending_false	<code>requests.findById(id).status=APPROVED</code>	reject	<code>false</code>	<code>verify(requests, never()).updateStatus(anyInt(), any(), any(), any())</code>
fulfillRequest	F1	fulfill_approved_stockOK_completed_true	<code>findById=APPROVED(type,qty,approvedBy=10) , inventory.getAvailable(type) >= qty</code>	fulfill	<code>true ; deduct + COMPLETED</code>	<code>verify(inventory).deduct(eq(type), eq(qty)); verify(requests).updateStatus(id, COMPLETED, 10, null)</code>
fulfillRequest	F2	fulfill_insufficient_stock_throw	<code>getAvailable(type) < qty</code>	fulfill	<code>throws IllegalStateException("Insufficient stock")</code>	<code>verify(inventory, never()).deduct(any(), anyInt()); verify(requests, never()).updateStatus(anyInt(), any(), any(), any())</code>
fulfillRequest	F3	fulfill_status_not_approved_false	<code>findById.status=PENDING</code>	fulfill	<code>false</code>	<code>verify(inventory, never()).deduct(any(), anyInt()); verify(requests, never()).updateStatus(anyInt(), any(), any(), any())</code>
fulfillRequest	F4	fulfill_stock_boundary_eq_true	<code>getAvailable(type) == qty</code>	fulfill	<code>true</code>	<code>verify(inventory).deduct(eq(type), eq(qty)); verify(requests).updateStatus(id, COMPLETED, any(), isNull())</code>
fulfillRequest	F5	fulfill_not_found_false	<code>requests.findById(id)=empty</code>	fulfill	<code>false</code>	<code>verify(inventory, never()).deduct(any(), anyInt()); verify(requests, never()).updateStatus(anyInt(), any(), any(), any())</code>

2) Thiết kế Test Cases (Ma trận)

notifyHospital	N1	notify_completed_with_email_true	<code>requests.findById(id).status=COMPLETED , hospitals.findEmailById(h)="hos@mail.com" , emailService.send= true , notificationService.pushToHospital= true</code>	notify	true	<code>verify(emailService).send(eq("hos@mail.com"), eq("Blood Request COMPLETED"), argThat(b -> b.contains("#"+id) && b.contains("COMPLETED"))); verify(notificationService).pushToHospital(eq(h), eq("Request "+id+" is COMPLETED"))</code>
notifyHospital	N2	notify_invalid_status_pending_false	<code>status=PENDING</code>	notify	false	<code>verify(hospitals, never()).findEmailById(anyInt()); verify(emailservice, never()).send(any(), any(), any()); verify(notificationService, never()).pushToHospital(anyInt(), any())</code>
notifyHospital	N3	notify_missing_email_false	<code>status=APPROVED , findEmailById(h)=empty</code>	notify	false	<code>verify(emailService, never()).send(any(), any(), any()); verify(notificationService, never()).pushToHospital(anyInt(), any())</code>
notifyHospital	N4	notify_partial_emailOk_notifFail_false	<code>send=true , push=false</code>	notify	false	<code>verify(emailService).send(...); verify(notificationService).pushToHospital(...); assertFalse</code>
notifyHospital	N5	notify_partial_emailFail_notifOk_false	<code>send=false , push=true</code>	notify	false	<code>verify(emailService).send(...); verify(notificationService).pushToHospital(...); assertFalse</code>
notifyHospital	N6	notify_both_fail_false	<code>send=false , push=false</code>	notify	false	<code>verify(emailService).send(...); verify(notificationService).pushToHospital(...); assertFalse</code>
notifyHospital	N7	notify_not_found_false	<code>requests.findById(id)=empty</code>	notify	false	<code>verify(hospitals, never()).findEmailById(anyInt()); verify(emailservice, never()).send(any(), any(), any()); verify(notificationService, never()).pushToHospital(anyInt(), any())</code>

3) SINH TEST CODE (JEST) – GENERATE TEST CODE

5 lớp test:

- BloodRequestServiceCreateTest
- BloodRequestServiceApproveTest
- BloodRequestServiceRejectTest
- BloodRequestServiceFulfillTest
- BloodRequestServiceNotifyTest

Mẫu cấu trúc test (chuẩn hóa)

- Sử dụng `@Mock` cho toàn bộ dependency và `@InjectMocks` cho `BloodRequestService`.
- **Given–When–Then** rõ ràng:
 - Given: stub when(...) cho repo/service.
 - When: gọi method service.
 - Then: assert... + verify(...) (có `never()` và `NoInteractions()` cho negative path).
- Dùng **biểu thức khớp tham số** (`eq`, `any`, `argThat`) để kiểm tra nội dung email/notification.
- Tận dụng **test biên**: `quantity = 0/1`, `tồn kho = quantity/quantity-1`.

4) DEBUG TESTS



- **Sai stub dẫn tới NPE:** luôn set up when(...).thenReturn(Optional.of(...)) cho findById.
- **Thứ tự tương tác:** nếu muốn chắc chắn quy trình, dùng InOrder để assert thứ tự ví dụ: findById → getAvailable → deduct → updateStatus.
- **Verify không mong muốn:** với negative path, dùng verify(requests, never()).updateStatus(...) và verifyNoMoreInteractions(requests) để tăng branch coverage.
- **Khả dụng kho:** kiểm tra rõ 3 nhánh <, ==, > (Bạn đã có case ==).
- **Kênh thông báo:** ở notifyHospital, thêm case “email true, notification false” và ngược lại để phủ nhánh e && n.

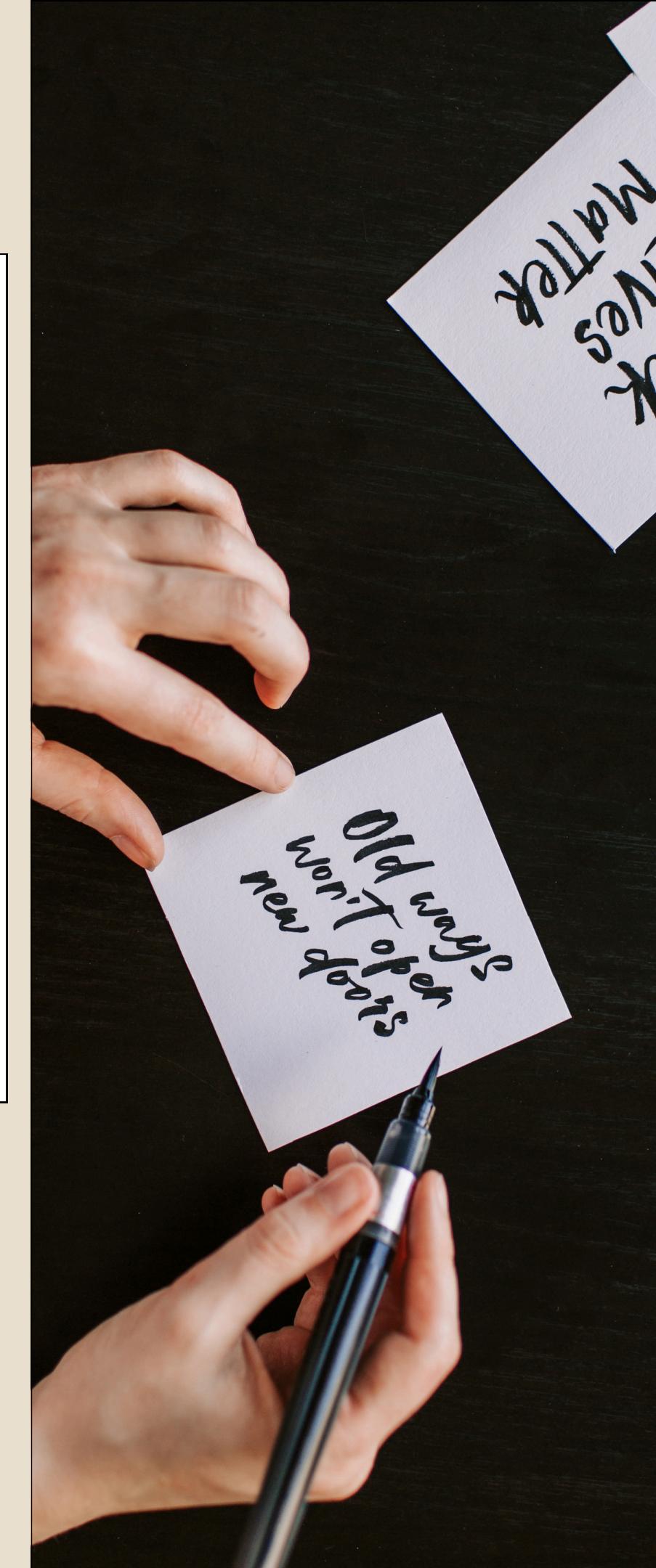
5) MOCKING & OPTIMIZATION

- **Mockito best practices**

1. Chỉ stub những gì test cần. Tránh over-stubbing để dễ đọc.
2. Dùng ArgumentCaptor khi cần kiểm tra object đã save/updateStatus với trường cụ thể.
3. Dùng verifyNoInteractions(...) để đảm bảo **early return** đúng (ví dụ admin invalid).

- **Tối ưu test**

4. Gom các biến lặp (ví dụ tạo BloodRequest mẫu) vào @BeforeEach.
5. Dùng **ParameterizedTest** cho các biến thế hợp lệ/không hợp lệ của bloodType, quantity.
6. Đặt tên test dạng: method_condition_expectedResult (bạn đã đặt rất tốt).



6) DOCUMENTATION & DEMO

TIÊU CHÍ HOÀN TẤT

- ≥ 15 test trải đều 5 hàm chính.
- Line & Branch coverage $\geq 80\%$
(JaCoCo fail build nếu dưới ngưỡng).
- Mọi dependency được mock (không cần triển khai thật DB/email/push).

DEMO FLOW

- Giới thiệu service & luồng trạng thái.
- Trình bày ma trận test (mục 2) \rightarrow nhấn vào các case biên quan trọng.
- Chạy mvn verify trực tiếp \rightarrow mở report JaCoCo.
- Minh họa 1–2 verify negative path (ví dụ: admin invalid \rightarrow no interactions).
- Kết thúc bằng phần coverage và lesson learned (mock đúng điểm, test rõ nhánh).