

FRAMEWORK MODELLING UNTUK INTEGRASI ECONOMIC ORDER QUANTITY PADA SISTEM STOK GUDANG BERBASIS JAVA SPRING DAN POSTGRESQL

Fransisco Viero¹, Raden Teddy Iswahyudi², Agung Mulyo Widodo³, Nizirwan Anwar⁴

^{1,2,3,4} Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Esa Unggul

Email: fransvierooo9@gmail.com, raden.teddy@esaunggul.ac.id, agung.mulyo@esaunggul.ac.id, nizirwan.anwar@esaunggul.ac.id

Abstrak

Pengelolaan stok membutuhkan konsistensi antara transaksi operasional dan keputusan kuantitas pemesanan. Penelitian ini mengembangkan prototipe sistem stok gudang berbasis Java Spring dan PostgreSQL dengan Economic Order Quantity (EOQ), serta mengusulkan framework modelling yang menautkan kebutuhan, model perilaku, model struktur, model data, arsitektur implementasi, dan bukti pengujian. Penelitian bersifat terapan dengan alur Waterfall yang mencakup analisis kebutuhan, perancangan, implementasi, validasi algoritma, dan black-box testing. EOQ dihitung dari kebutuhan tahunan (D), biaya pemesanan per pesanan (S), dan biaya penyimpanan per unit per tahun (H). Evaluasi terkontrol menggunakan lima item simulasi dan 35 skenario pengujian. Rekonsiliasi stok menghasilkan saldo hitung 575 unit yang sama dengan saldo sistem. Perbandingan EOQ manual dan aplikasi menunjukkan seluruh selisih absolut di bawah toleransi 0,01 unit, dengan selisih terbesar 0,0045 unit. Tiga ketidaksesuaian awal terkait duplikasi kode, pengeluaran melebihi stok, dan $H = 0$ diperbaiki; seluruh 35 skenario lulus pada pengujian ulang. Kontribusi utama penelitian terletak pada keterlacakan kebutuhan-ke-model-ke-implementasi-ke-bukti, bukan pada modifikasi rumus EOQ. Karena evaluasi menggunakan data simulasi, hasil belum dapat digunakan untuk mengklaim penghematan biaya operasional aktual.

Kata kunci: Economic Order Quantity, framework modelling, Java Spring, PostgreSQL, stok gudang; traceability.

Abstract

Inventory control requires consistency between operational transactions and order-quantity decisions. This study develops a warehouse stock prototype using Java Spring and PostgreSQL with Economic Order Quantity (EOQ) and proposes a modelling framework that links requirements, behavioural models, structural models, data models, implementation architecture, and testing evidence. The study follows an applied Waterfall-oriented process covering requirements analysis, design, implementation, algorithm validation, and black-box testing. EOQ is computed from annual demand (D), ordering cost per order (S), and annual holding cost per unit (H). Controlled evaluation uses five simulated items and 35 functional test scenarios. Stock reconciliation produced a calculated balance of 575 units, identical to the system balance. Manual-to-application EOQ comparison showed absolute differences below the 0.01-unit tolerance, with a maximum difference of 0.0045 units. Three initial defects concerning duplicate item codes, issues exceeding available stock, and $H = 0$ were corrected; all 35 scenarios passed the final retest. The main contribution is requirement-to-model-to-implementation-to-evidence traceability rather than a new EOQ formula. Because the evaluation relies on simulated data, the results do not establish actual operational cost savings.

Keywords: Economic Order Quantity, framework modelling, inventory,; Java Spring,; PostgreSQL, traceability.

1. Pendahuluan

Perkembangan sistem informasi mendorong organisasi mengelola persediaan tidak lagi sebagai kumpulan catatan yang berdiri sendiri, melainkan sebagai aliran data yang harus konsisten dari penerimaan, penyimpanan, pengeluaran, hingga pelaporan. Kualitas informasi persediaan bergantung pada ketepatan identitas barang, jumlah, waktu transaksi, pihak yang terlibat, dan keterlacakan sumber perubahan saldo. Dalam konteks ini, basis data terintegrasi bukan sekadar media penyimpanan. Basis data menjadi mekanisme pengendalian agar setiap

perubahan stok mempunyai asal transaksi yang dapat diperiksa kembali [1] [2].

Permasalahan umum pada pengelolaan gudang muncul ketika barang masuk, barang keluar, saldo, dan laporan dikelola pada media berbeda. Pola tersebut meningkatkan peluang duplikasi, keterlambatan pembaruan, perbedaan format, dan kesulitan rekonsiliasi. Wau [5] menunjukkan bahwa digitalisasi persediaan dapat mempercepat penyediaan informasi transaksi dan laporan. Apriyanti dan Bernanda [3] juga menemukan bahwa sistem inventori dapat membantu menata transaksi dan otomatisasi laporan. Mulyana [4] menekankan pentingnya pemodelan objek untuk

mengurangi selisih antara catatan dan barang fisik. Temuan-temuan tersebut memperlihatkan bahwa integrasi data merupakan syarat dasar sebelum sistem digunakan untuk mendukung keputusan persediaan.

Di luar pencatatan, pengelola gudang menghadapi keputusan mengenai berapa banyak barang yang sebaiknya dipesan. Pemesanan terlalu kecil meningkatkan frekuensi pemesanan dan risiko kekurangan, sedangkan pemesanan terlalu besar menambah rata-rata persediaan serta biaya penyimpanan. Economic Order Quantity (EOQ) memberikan pendekatan kuantitatif untuk menyeimbangkan biaya pemesanan dan penyimpanan pada asumsi tertentu [6]. Kajian Alnahhal et al. [7] menegaskan bahwa EOQ tetap relevan sebagai model dasar, tetapi penerapannya harus menyatakan batas asumsi karena ketidakpastian rantai pasok modern tidak seluruhnya ditangkap oleh model klasik.

Integrasi EOQ ke dalam sistem informasi persediaan telah dibahas dalam penelitian terdahulu. Setyadi et al. [9] menerapkan EOQ dan reorder point untuk menghasilkan informasi pengendalian persediaan, sedangkan penelitian sistem gudang lain lebih menonjolkan integrasi transaksi, pelaporan, dan tahapan pengembangan [5] [8]. Celah yang masih penting adalah bagaimana artefak analisis, model sistem, struktur data, algoritma, implementasi, dan pengujian dihubungkan sehingga klaim hasil dapat ditelusuri sampai bukti pengujiannya. Tanpa keterlacakan, diagram dapat menjadi dekorasi dokumentasi, sementara pengujian hanya mengikuti menu tanpa menunjukkan hubungan dengan kebutuhan.

Penelitian ini menempatkan framework modelling sebagai pengikat antara kebutuhan, model perilaku, model struktur, model data, arsitektur implementasi, algoritma keputusan, dan bukti pengujian. Use case digunakan untuk menetapkan layanan, activity diagram untuk aliran dan keputusan, class diagram untuk objek dan tanggung jawab, serta entity relationship diagram (ERD) untuk struktur data. Arsitektur Java Spring memetakan tanggung jawab ke controller, service, repository, dan entity, sedangkan PostgreSQL menjaga relasi serta transaksi data. Setiap kebutuhan kemudian dipasangkan dengan skenario black-box sehingga jalur dari kebutuhan ke hasil dapat diaudit [10] [11] [12].

Kontribusi penelitian bukan pengembangan rumus EOQ baru. Kontribusinya adalah rancangan integratif yang menjadikan EOQ sebagai layanan yang memakai parameter terdokumentasi, menyimpan riwayat perhitungan, dan berada dalam alur pengelolaan stok yang dapat diuji. Pendekatan ini menghindari dua jebakan yang sering muncul pada penelitian rancang bangun: menganggap keberadaan antarmuka sebagai bukti keberhasilan, dan menganggap angka EOQ sebagai keputusan final tanpa menampilkan parameter serta asumsi. Hasil EOQ diposisikan sebagai

rekomendasi analitis yang tetap perlu dibaca bersama stok tersedia, lead time, kapasitas, kemasan, dan kebijakan pemasok [13] [7]. Berdasarkan latar tersebut, tujuan penelitian adalah (1) merancang framework modelling yang menjaga konsistensi antara kebutuhan, model, implementasi, dan pengujian; (2) mengintegrasikan transaksi barang masuk-keluar dengan posisi stok pada satu basis data; (3) menerapkan dan memvalidasi EOQ pada layanan aplikasi; dan (4) mengevaluasi fungsi prototipe melalui black-box testing. Karena naskah sumber menggunakan data dummy pada evaluasi, artikel ini secara eksplisit membatasi klaim pada validitas implementasi dan perilaku prototipe di lingkungan uji, bukan pada peningkatan efisiensi operasional di organisasi nyata.

Tabel 1. Posisi penelitian terhadap studi terdahulu

Studi	Fokus utama	Keterbatasan relatif terhadap penelitian ini
Wau (2022)	Sistem persediaan berbasis web dan Waterfall	Tidak mengintegrasikan EOQ serta keterlacakan kebutuhan-ke-skenario uji.
Apriyanti & Bernanda (2023)	Inventori, laporan otomatis, black-box	Fokus pada pendataan barang, bukan layanan rekomendasi EOQ.
Setyadi et al. (2024)	EOQ dan reorder point dalam sistem persediaan	Menekankan metode pengendalian; penelitian ini menajamkan framework modelling dan bukti traceability pada Java Spring-PostgreSQL.
Melisa et al. (2025)	Sistem gudang berbasis web dengan Waterfall	Tidak menempatkan EOQ sebagai layanan keputusan terintegrasi.

2. Metode Penelitian

2.1 Desain penelitian dan alur pengembangan

Penelitian menggunakan pendekatan terapan melalui pengembangan sistem. Artefak yang dinilai meliputi spesifikasi kebutuhan, model, struktur data, implementasi aplikasi, layanan EOQ, dan bukti pengujian. Alur pengembangan mengikuti Waterfall sebagai kerangka dokumentasi: analisis kebutuhan, perancangan, implementasi, pengujian, perbaikan, dan uji ulang. Waterfall dipilih karena ruang lingkup fungsi telah dibatasi pada autentikasi, data induk, transaksi, stok, EOQ, serta laporan. Umpan balik terbatas tetap dilakukan ketika pengujian menemukan ketidaksesuaian [14] [12].

Informasi kebutuhan secara metodologis dihimpun melalui observasi proses, wawancara semiterstruktur, dan pemeriksaan dokumen. Data yang dibutuhkan mencakup alur transaksi, aktor, atribut barang dan

pemasok, format dokumen, aturan perubahan stok, kebutuhan laporan, serta parameter EOQ. Data kuantitatif EOQ terdiri atas D, S, dan H. Namun, evaluasi numerik pada naskah sumber menggunakan data simulasi; karena itu, artikel ini tidak menyatakan angka tersebut sebagai data lapangan. Perbedaan ini penting agar validasi rumus tidak disalahartikan sebagai evaluasi manfaat operasional.

Keabsahan rancangan diarahkan melalui triangulasi teknik dan keterlacakan artefak. Kebutuhan yang diperoleh dari proses bisnis diterjemahkan menjadi fungsi berkode. Setiap fungsi harus mempunyai model perilaku atau struktur yang sesuai, komponen implementasi yang menjalankan aturan, serta skenario pengujian yang memverifikasi perilaku. Dengan cara ini, perubahan pada satu artefak dapat diperiksa dampaknya pada artefak lain.

2.2 Framework modelling dan traceability

Framework modelling dibentuk sebagai enam lapisan yang saling terhubung. Lapisan masalah dan kebutuhan mendefinisikan tujuan serta aturan bisnis. Lapisan perilaku memodelkan interaksi dan aliran. Lapisan struktur memodelkan objek dan relasi. Lapisan data mendefinisikan tabel, kunci, dan integritas. Lapisan implementasi memetakan logika ke arsitektur berlapis Java Spring dan PostgreSQL. Lapisan verifikasi menghubungkan kebutuhan dengan skenario black-box dan validasi numerik EOQ. Konsistensi nama konsep seperti Barang, Pemasok, Barang Masuk, Barang Keluar, Stok, Pengguna, dan Hasil EOQ dipertahankan lintas artefak agar tidak muncul entitas atau aturan yang kehilangan pasangan implementasi dan bukti uji [10] [11].

Tabel 2. Framework modelling dan bukti keterlacakan

Lapisan	Artefak inti	Pertanyaan kendali	Bukti
Kebutuhan	F-01 s.d. F-08, aturan bisnis	Apa fungsi dan batas yang wajib dipenuhi?	Matriks kebutuhan dan aturan.
Perilaku	Use case dan activity diagram	Bagaimana aktor berinteraksi dan keputusan terjadi?	Alur utama, alternatif, validasi.
Struktur	Class diagram	Objek apa yang menyimpan data dan menjalankan tanggung jawab?	Entity/service/controller yang konsisten.

Data	ERD dan skema PostgreS QL	Bagaimana relasi serta integritas dipertahankan?	PK, FK, UNIQUE, CHECK, transaksi.
Implementasi	Controller -service-repository	Di mana aturan stok dan EOQ dijalankan?	Pemisahan lapisan dan transaction boundary.
Verifikasi	PB-01 s.d. PB-35 dan validasi EOQ	Bukti apa yang menyatakan fungsi sesuai?	Hasil aktual, status, defect, retest.

2.3 Model EOQ dan aturan perhitungan

EOQ menggunakan tiga parameter: D adalah kebutuhan tahunan, S adalah biaya setiap kali pemesanan, dan H adalah biaya penyimpanan satu unit per tahun. Ketiganya harus numerik, lebih besar dari nol, konsisten periode dan satuan, serta memiliki sumber yang dapat ditelusuri. Nilai kosong, negatif, teks, atau $H = 0$ ditolak sebelum proses hitung. Apabila D diturunkan dari periode yang lebih pendek dari satu tahun, metode annualisasi harus terdokumentasi dan hanya digunakan bila periode representatif [6].

$$EOQ = Q^* = \sqrt{(2DS/H)}$$

$$F = D/Q^*$$

$$TIC(Q) = (D/Q)S + (Q/2)H$$

Aplikasi menghitung nilai EOQ mentah, frekuensi pemesanan tahunan (F), dan total biaya persediaan relevan (TIC). Nilai mentah dipertahankan untuk validasi, sedangkan pembulatan diterapkan pada tampilan atau rekomendasi operasional. Hasil sistem dibandingkan dengan perhitungan manual menggunakan parameter identik. Selisih absolut ditetapkan sebagai $|EOQ_{sistem} - EOQ_{manual}|$ dan toleransi validasi sebesar 0,01 unit untuk memisahkan perbedaan presentasi angka dari kesalahan algoritma. Analisis sensitivitas tambahan memvariasikan D, S, dan H sebesar $\pm 20\%$ untuk menunjukkan arah perubahan rekomendasi.

2.4 Implementasi dan pengujian

Aplikasi menggunakan arsitektur berlapis. Controller menerima permintaan dan mengelola aliran HTTP; service memegang aturan bisnis transaksi dan EOQ; repository menyediakan akses data melalui Spring Data JPA; entity merepresentasikan struktur domain; PostgreSQL menyimpan data persisten. Perubahan stok dan dokumen transaksi dibungkus dalam transaksi basis data agar kegagalan pada detail

tidak meninggalkan saldo parsial. Pada barang keluar, sistem memeriksa ketersediaan sebelum commit sehingga stok negatif dapat dicegah [15] [16] [17] [18].

Pengujian black-box diturunkan dari kebutuhan fungsional dan mencakup kelas ekuivalensi, nilai batas, serta transisi sederhana. Tiga kelompok skenario digunakan: 11 skenario login dan data induk, 14 skenario transaksi dan stok, serta 10 skenario EOQ dan laporan. Status lulus diberikan hanya bila keluaran aktual sama dengan hasil yang diharapkan dan perubahan data juga konsisten. Ketidaksesuaian dicatat sebagai defect, diperbaiki, lalu skenario terkait dijalankan ulang. Prinsip ini sejalan dengan ISO/IEC/IEEE 29119-1:2022 yang menekankan dokumentasi kondisi, masukan, keluaran yang diharapkan, keluaran aktual, dan status pengujian [19] [20].

3. Hasil dan Pembahasan

3.1 Hasil framework modelling dan integrasi transaksi-stok

Pemetaan kebutuhan menghasilkan delapan kelompok fungsi: autentikasi, data barang, pemasok, barang masuk, barang keluar, stok, EOQ, dan laporan. Setiap kelompok dihubungkan dengan rentang skenario uji tertentu. Barang masuk harus menyimpan dokumen, detail, dan penambahan stok secara atomik. Barang keluar menambahkan syarat jumlah positif, barang tersedia, dan saldo mencukupi. EOQ mewajibkan D, S, $H > 0$ serta konsistensi satuan dan periode. Laporan memverifikasi bahwa filter, baris, total, dan hasil ekspor berasal dari sumber data yang sama.

Hasil modelling menempatkan service transaksi sebagai satu-satunya jalur perubahan saldo. Desain ini penting karena saldo yang dapat diubah langsung dari antarmuka atau query terpisah akan memutus jejak transaksi. Pada penerimaan, sistem memvalidasi pengguna, nomor dokumen, tanggal, item, dan kuantitas sebelum menyimpan transaksi dan memperbarui stok. Pada pengeluaran, pemeriksaan saldo dilakukan dalam transaction boundary yang sama. PostgreSQL memperkuat aturan tersebut melalui primary key, foreign key, UNIQUE, CHECK, dan sifat transaksi ACID. Struktur ini membuat rekonsiliasi dapat dilakukan dari saldo menuju riwayat mutasi dan dokumen sumber [2].

Dalam simulasi lima barang, total saldo awal adalah 410 unit. Barang masuk sebesar 250 unit dan barang keluar sebesar 85 unit menghasilkan saldo hitung 575 unit. Saldo sistem juga 575 unit, sehingga selisih rekonsiliasi semua item bernilai 0. Hasil tersebut tidak membuktikan ketepatan untuk seluruh kombinasi transaksi, tetapi menunjukkan bahwa aturan aritmetis dan penyimpanan terintegrasi konsisten pada

data dan skenario yang diuji. Pengujian kasus rollback, jumlah nol, barang tidak ada, dan dua permintaan yang bersaing memberikan konteks lebih kuat dibanding hanya menampilkan saldo akhir.

Dari sisi framework modelling, temuan pentingnya adalah keterlaksanaan dua arah. Kebutuhan “pengeluaran tidak boleh menyebabkan stok negatif” diterjemahkan ke percabangan pada activity flow, aturan service, transaction boundary, dan skenario PB-19/PB-21. Sebaliknya, defect pada skenario dapat ditelusuri kembali ke aturan yang kurang lengkap pada implementasi. Keterlaksanaan semacam ini membuat model berfungsi sebagai alat pengendalian desain, bukan sekadar dokumentasi pascaimplementasi.

3.2 Validasi EOQ dan sensitivitas parameter

Validasi menggunakan lima item simulasi dengan parameter berbeda. Tujuannya bukan mewakili pola permintaan lapangan, melainkan memeriksa apakah layanan EOQ konsisten terhadap variasi nilai masukan. Seluruh hasil aplikasi berada dalam toleransi 0,01 unit terhadap perhitungan manual. Selisih terbesar 0,0045 unit terjadi pada Barang B akibat pembulatan tampilan. Karena perbandingan dilakukan dengan parameter identik dan nilai manual tanpa pembulatan, hasil ini mendukung klaim bahwa rumus dan kebijakan presisi diterapkan secara konsisten pada kasus uji.

Tabel 3. Validasi EOQ manual dan keluaran sistem pada data simulasi

Item	D (unit/tahun)	S (Rp/pesan)	H (Rp/unit/tahun)	EOQ manual	EOQ sistem	Selisih	Status
A	12.000	50.000	2.000	774,5967	774,60	0,0033	Lulus
B	8.000	40.000	1.600	632,4555	632,46	0,0045	Lulus
C	15.000	75.000	2.500	948,6833	948,68	0,0033	Lulus
D	6.000	30.000	1.200	547,7226	547,72	0,0026	Lulus
E	20.000	60.000	3.000	894,4272	894,43	0,0028	Lulus

Nilai EOQ tidak dapat diinterpretasikan hanya dari tingkat kebutuhan. Item E mempunyai D tertinggi, tetapi kuantitas ekonomis juga dipengaruhi S dan H. Hubungan akar kuadrat menjelaskan mengapa perubahan input tidak diterjemahkan secara linear. Pada data dasar D = 12.000, S = Rp50.000, dan H = Rp2.000, EOQ adalah 774,60 unit. Kenaikan D sebesar 20% meningkatkan EOQ menjadi 848,53 unit atau

sekitar 9,54%; kenaikan S sebesar 20% memberikan perubahan yang sama. Sebaliknya, kenaikan H sebesar 20% menurunkan EOQ menjadi 707,11 unit atau sekitar 8,71%. Ketika H turun 20%, EOQ meningkat menjadi 866,03 unit atau sekitar 11,80%.

Analisis sensitivitas memperkuat dua hal. Pertama, implementasi harus menampilkan parameter bersama hasil karena rekomendasi dapat bergeser ketika asumsi biaya berubah. Kedua, presisi komputasi tidak sama dengan akurasi keputusan. Sistem dapat menghitung rumus secara tepat, tetapi bila S atau H berasal dari estimasi yang lemah, rekomendasi tetap berpotensi bias. Karena itu, riwayat EOQ menyimpan barang, periode, D, S, H, hasil, waktu, dan pengguna. Transparansi input membuat pengguna dapat membedakan kesalahan data, perubahan asumsi, dan perubahan algoritma [21] [7].

Hasil ini sejalan dengan Setyadi et al. [9] bahwa model pengendalian dapat diintegrasikan ke sistem informasi. Perbedaannya, fokus penelitian ini bukan memperluas metode ke safety stock dan reorder point, melainkan memastikan jejak dari parameter ke hasil, aturan validasi, dan skenario verifikasi. Strategi tersebut sesuai untuk konteks prototipe yang menempatkan EOQ sebagai rekomendasi kuantitas, bukan sebagai mesin keputusan pembelian otomatis.

3.3 Hasil black-box testing dan perbaikan defect

Pengujian awal menemukan tiga ketidaksesuaian yang memiliki konsekuensi langsung terhadap integritas data dan ketepatan algoritma. Defect pertama memungkinkan kode barang duplikat. Perbaikannya menambahkan kendala unik dan penanganan pelanggaran constraint. Defect kedua memungkinkan pengeluaran melebihi stok sehingga saldo berpotensi negatif. Perbaikan dilakukan melalui validasi saldo dan penguncian data dalam transaction boundary. Defect ketiga terjadi ketika $H = 0$ memicu risiko pembagian nol pada layanan EOQ; perbaikan menambahkan validasi $H > 0$ sebelum perhitungan. Ketiga skenario kemudian lulus pada uji ulang.

Tabel 4. Ringkasan hasil black-box testing setelah perbaikan dan uji ulang

Kelompok fungsi	Jumlah skenario	Lulus	Gagal	Terhalang	Belum diuji
Login dan data induk	11	11	0	0	0
Transaksi dan stok	14	14	0	0	0
EOQ dan laporan	10	10	0	0	0
Total	35	35	0	0	0

Pada kondisi akhir simulasi, seluruh 35 skenario berstatus lulus. Pengujian transaksi mencakup penerimaan multi-item, jumlah nol, nomor dokumen duplikat, rollback ketika detail gagal, pengeluaran tepat sebesar stok, pengeluaran melebihi stok, kuantitas negatif, penggunaan barang yang tidak ada, riwayat mutasi, serta rekonsiliasi. Pengujian EOQ mencakup nilai valid, D nol/negatif, S kosong atau teks, $H = 0$, nilai desimal, penyimpanan riwayat, dan pelaporan. Laporan juga diuji pada periode tanpa data, periode dengan data, filter item, dan ekspor PDF.

Persentase kelulusan 100% tidak boleh dibaca sebagai pernyataan bahwa perangkat lunak bebas kesalahan. Black-box testing hanya memberi bukti bahwa perilaku yang diperiksa sesuai dengan spesifikasi pada lingkungan dan data tertentu. Standar pengujian menekankan bahwa kualitas kesimpulan bergantung pada cakupan skenario, dokumentasi hasil aktual, dan keterulangan prosedur [19]. Temuan tiga defect awal justru memperlihatkan nilai framework modelling: skenario bukan formalitas, melainkan mekanisme untuk menemukan ketidaksesuaian antara kebutuhan, aturan implementasi, dan data.

3.4 Pembahasan kontribusi dan implikasi

Kontribusi pertama adalah integrasi transaksi dan saldo melalui pola arsitektur yang menempatkan service sebagai penjaga aturan bisnis dan PostgreSQL sebagai lapisan integritas data. Pendekatan ini memperkuat praktik sistem persediaan yang telah ditunjukkan Wau [5], Apriyanti dan Bernanda [3], serta Melisa et al. [8], tetapi menambahkan fokus pada keterlacakan perubahan saldo. Informasi persediaan menjadi lebih dapat diaudit karena setiap perubahan diarahkan melalui dokumen transaksi dan pengguna pencatat.

Kontribusi kedua adalah pemisahan antara “ketepatan perhitungan” dan “ketepatan keputusan”. Sistem memvalidasi rumus EOQ, frekuensi, dan TIC, tetapi tidak mengklaim bahwa nilai tersebut otomatis merupakan jumlah pembelian final. EOQ dasar mengasumsikan permintaan relatif stabil, biaya dapat ditentukan, dan penerimaan terjadi sekaligus. Safety stock, reorder point, diskon kuantitas, ketidakpastian lead time, barang rusak, kapasitas gudang, dan hubungan antaritem belum tercakup. Posisi ini konsisten dengan Alnahhal et al. [7], yang menunjukkan bahwa model EOQ modern berkembang untuk menangani ketidakpastian yang lebih kompleks.

Kontribusi ketiga, dan yang paling membedakan artikel ini, adalah framework modelling sebagai rantai bukti. Pada banyak naskah rancang bangun, use case, activity, class diagram, ERD, arsitektur, dan pengujian dipresentasikan sebagai artefak terpisah. Di sini, nama konsep dan aturan dijaga lintas artefak, lalu dikunci

dengan kode kebutuhan dan kode skenario uji. Sebuah aturan bisnis dianggap lengkap hanya bila dapat ditemukan pada kebutuhan, model, implementasi, dan bukti pengujian. Pendekatan tersebut memberi bahasa evaluasi yang lebih tajam untuk penelitian pengembangan sistem karena reviewer dapat memeriksa jalur klaim tanpa bergantung pada tangkapan layar aplikasi.

Secara praktis, framework ini dapat digunakan sebagai checklist pengembangan. Jika use case menghasilkan “Hasil EOQ”, maka harus ada kelas atau entitas untuk menampung parameter dan hasil; harus ada service yang menjalankan validasi dan perhitungan; harus ada tabel basis data bila riwayat disimpan; dan harus ada skenario yang memeriksa nilai valid serta tidak valid. Jika satu mata rantai hilang, desain perlu direvisi. Dengan demikian, modelling berfungsi sebagai kompas yang menjaga artefak tidak hanyut menjadi pulau-pulau dokumentasi.

Implikasi manajerialnya tetap bersifat terbatas. Prototipe menyediakan dasar kuantitatif yang transparan untuk jumlah pemesanan dan menurunkan ketergantungan pada perkiraan semata, tetapi keputusan pembelian tetap perlu mempertimbangkan stok tersedia, lead time, kapasitas, kebijakan pemasok, dan kualitas data biaya. Tanpa data lapangan sebelumnya, penelitian ini tidak dapat menyatakan penghematan waktu atau biaya. Klaim yang didukung hanya mencakup konsistensi transaksi pada skenario uji, ketepatan implementasi EOQ terhadap acuan, dan kesesuaian fungsi prototipe setelah perbaikan.

4. Keterbatasan Penelitian dan Agenda Lanjutan

Keterbatasan utama adalah penggunaan data dummy pada rekonsiliasi stok, perhitungan EOQ, serta pengujian black-box. Data tersebut cukup untuk menguji alur dan aturan, tetapi tidak mewakili variasi permintaan, pola musiman, kesalahan pencatatan nyata, atau perubahan biaya pada organisasi. Oleh karena itu, nilai EOQ pada Tabel 3 tidak boleh digunakan sebagai dasar pembelian aktual. Studi lanjutan perlu mengganti parameter simulasi dengan data historis yang mendapat izin, mendokumentasikan cara pembentukan D, S, dan H, dan mengulang validasi pada beberapa periode.

Keterbatasan kedua berasal dari model EOQ dasar. Penelitian belum memasukkan safety stock, reorder point, permintaan probabilistik, diskon kuantitas, umur simpan, ataupun multi-gudang. Penambahan fitur seharusnya tidak dilakukan sebagai tempelan menu.

Setiap perluasan perlu melewati framework modelling yang sama: kebutuhan baru, perubahan model perilaku, struktur data, layanan, dan skenario uji. Untuk barang dengan pola musiman, EOQ dapat dikombinasikan dengan peramalan atau dibandingkan dengan kebijakan persediaan alternatif.

Keterbatasan ketiga adalah cakupan kualitas perangkat lunak. Black-box fungsional belum mengukur keamanan mendalam, kinerja beban tinggi, reliabilitas jangka panjang, pencadangan-pemulihan, portabilitas, atau usability terukur. ISO/IEC 25010:2023 memisahkan berbagai karakteristik kualitas tersebut, sehingga penelitian berikutnya sebaiknya menambahkan pengujian kinerja, keamanan, konkurensi, usability, dan penerimaan pengguna [22]. Evaluasi dampak juga perlu membandingkan ketepatan stok, kejadian stockout, rata-rata nilai persediaan, waktu kerja, dan total biaya sebelum serta sesudah implementasi.

5. Kesimpulan

Penelitian menghasilkan prototipe sistem pengelolaan stok gudang berbasis Java Spring dan PostgreSQL yang mengintegrasikan transaksi, posisi stok, layanan EOQ, riwayat perhitungan, dan laporan. Framework modelling menghubungkan kebutuhan, use case, activity flow, struktur kelas, ERD, arsitektur berlapis, dan skenario uji sebagai satu rantai keterlacakan. Pada data simulasi lima item, rekonsiliasi saldo awal 410 unit, barang masuk 250 unit, dan barang keluar 85 unit menghasilkan saldo 575 unit yang sama dengan saldo sistem. Validasi EOQ menunjukkan seluruh hasil aplikasi berada dalam toleransi 0,01 unit terhadap perhitungan manual, dengan selisih maksimum 0,0045 unit. Pengujian black-box menemukan tiga defect awal yang kemudian diperbaiki, dan seluruh 35 skenario lulus pada kondisi akhir simulasi. Hasil tersebut mendukung kesesuaian implementasi pada ruang lingkup yang diuji, tetapi belum membuktikan efisiensi biaya operasional karena data evaluasi masih bersifat dummy. Untuk meningkatkan kekuatan eksternal dan kelayakan publikasi empiris, penelitian lanjutan perlu menggunakan data lapangan, menilai dampak sebelumnya, dan memperluas model persediaan serta karakteristik kualitas perangkat lunak.

Daftar

Pustaka