BAB III
METODE PENELITIAN

3.1 Objek Penelitian

Penelitian ini lahir dari permasalahan nyata yang ditemukan di lapangan, yakni bagaimana caranya mengelola stok *spare part* secara lebih cerdas agar perusahaan tidak lagi terjebak di antara dua kondisi yang sama-sama merugikan: gudang yang penuh barang tidak terpakai di satu sisi, dan mesin yang terpaksa berhenti karena komponen kritis kehabisan stok di sisi lain. Objek yang diteliti adalah sistem pengelolaan persediaan *spare part* di PT XYZ (nama disamarkan), sebuah perusahaan manufaktur yang berlokasi di kawasan industri Karawang. Penelitian ini dilaksanakan selama empat bulan, mulai dari tahap studi literatur hingga penyusunan laporan akhir.

Untuk membangun sistem prediksi yang handal, dibutuhkan tiga jenis data utama yang saling melengkapi. Data transaksi pemakaian *spare part* merekam pola konsumsi komponen dari waktu ke waktu. Data inventaris gudang memberikan gambaran kondisi stok yang tersedia saat ini. Sementara data riwayat *Purchase Order* digunakan untuk menghitung berapa lama waktu tunggu (*lead time*) sejak pemesanan dilakukan hingga barang tiba di gudang. Ketiga sumber data ini seluruhnya diperoleh dari sistem informasi internal perusahaan melalui proses ekspor data yang dikoordinasikan bersama bagian IT dan gudang perusahaan.

Dalam kerangka pemodelan *machine learning*, penelitian ini bekerja dengan dua kelompok variabel:

1. Variabel target (dependen): level stok minimum dan maksimum untuk setiap item *spare part*. Inilah nilai yang ingin dihasilkan oleh model bukan kategori, melainkan angka prediksi, sehingga pendekatannya adalah regresi.
2. Variabel input (independen): sejumlah faktor yang memengaruhi kebutuhan stok, antara lain nilai MTBF, frekuensi transaksi, rata-rata kuantitas pemakaian, koefisien variasi, dan kategori *movement* komponen. Variabel tanggal dan kode material digunakan sebagai kunci untuk merekayasa fitur-fitur tambahan yang lebih informatif.

Adapun batasan dalam penelitian ini adalah sebagai berikut. Pertama, data yang digunakan hanya berasal dari satu perusahaan manufaktur sehingga hasil

penelitian tidak serta-merta dapat digeneralisasi ke perusahaan lain dengan karakteristik berbeda. Kedua, penelitian ini hanya berfokus pada prediksi level stok minimum dan maksimum, dan tidak mencakup proses pengadaan atau negosiasi dengan *supplier* secara menyeluruh. Ketiga, model yang dibangun menggunakan data historis yang tersedia di sistem informasi internal, sehingga kualitas prediksi sangat bergantung pada kelengkapan dan akurasi data tersebut.

3.2 Prosedur Penelitian

Bagian ini menjabarkan langkah demi langkah yang ditempuh selama penelitian berlangsung. Tahapan disusun secara sistematis agar alur kerja mudah dipahami dan dapat direplikasi oleh peneliti lain yang memiliki ketertarikan pada topik yang sama. Secara garis besar, prosedur penelitian mencakup tiga hal utama: persiapan kebutuhan data dan perangkat, proses pengolahan data dari mentah hingga siap dimodelkan, dan skenario pengujian yang digunakan untuk memvalidasi sistem yang dihasilkan.

3.2.1 Kebutuhan Data dan Perangkat

Sebelum masuk ke proses teknis, perlu dijelaskan terlebih dahulu apa saja bahan dan alat yang dibutuhkan dalam penelitian ini.

1. Data

   Data yang digunakan bersifat kuantitatif dengan dimensi waktu dan item, artinya setiap catatan menyimpan informasi kapan transaksi terjadi dan komponen apa yang terlibat. Ketiga jenis data yang dibutuhkan seluruhnya bersumber dari sistem informasi internal perusahaan dan diperoleh melalui proses ekspor data resmi yang dikoordinasikan bersama tim IT dan bagian gudang PT XYZ. Rincian jenis data yang digunakan disajikan pada Tabel 3.1, Tabel 3.2, dan Tabel 3.3

Tabel 3.1 Dataset Inventaris *Spare Part*

No	Nama Variabel	Tipe Data	Deskripsi & Nilai
1	No	*Integer*	Nomor urut baris dalam laporan.

No	Nama Variabel	Tipe Data	Deskripsi & Nilai
2	Sloc	*Integer*	Kode lokasi gudang (storage location). Nilai: 1114 = Elektrik A, 1210 = Elektrik B, 1115 = Mekanik A, 1211 = Mekanik B.
3	Kode Barang	*String*	ID unik item spare part dalam format XXX-XXXXXX (mis. 143-005282). Digunakan sebagai kunci relasi ke tabel mutasi dan PO.
4	Nama Barang	*String*	Nama deskriptif item spare part (mis. BEARING NA4902).
5	Jumlah	*Integer*	Stok aktual item di gudang pada saat laporan dibuat. Digunakan sebagai Current_Stock dalam perhitungan Order_Qty = Max_Stock − Current_Stock.
6	Satuan	*String*	Satuan unit item. Nilai: BH, PCS, PC, UNIT, UNT, SET, L, KG, M, BTG, BT, ROL, TUB, CAN, TBG, DOS.
7	Nomor Rak	*String*	Lokasi fisik item dalam gudang dalam format [Baris].[Kolom]-[Level] (mis. C.3-2).
8	Mutasi Terakhir	*Date*	Tanggal transaksi terakhir (keluar/masuk) untuk item ini. Format: DD/MM/YYYY. Digunakan untuk mendeteksi item non-moving.
9	Status Stok	*String*	Status level stok otomatis. Nilai: safe = stok ≥ minimum, minimum stok = stok tepat di batas minimum, below minimum = stok di bawah batas minimum.
10	Harga Satuan	*Currency*	Harga per unit item dalam Rupiah (IDR). Bernilai 0 jika harga belum diinput.
11	Harga Total	*Currency*	Jumlah × Harga Satuan dalam Rupiah. Bernilai 0 jika harga satuan 0.
12	Stock Opname	*String*	Catatan hasil stock opname fisik. Nilai "-" jika belum dilakukan stock opname.

Tabel 3. 2 Dataset Mutasi Barang

No	Nama Variabel	Tipe Data	Deskripsi & Nilai
1	No	*Integer*	Nomor urut baris dalam laporan.
2	Kode Barang	*String*	ID unik item spare part. Kunci relasi ke tabel inventaris dan PO.
3	Nama Barang	*String*	Nama deskriptif item spare part.
4	Jumlah	*Integer*	Kuantitas barang pada transaksi ini. Untuk Keterangan = pemakaian / transfer barang keluar: jumlah yang keluar dari gudang. Untuk barang datang / transfer barang datang: jumlah yang masuk ke gudang. Kolom ini menjadi dasar seluruh fitur statistik pemakaian model (Mean_Usage, Daily_Usage, MTBF, dll.).
5	Stok	*Integer*	Sisa stok item di gudang setelah transaksi ini dicatat.
6	Satuan	*String*	Satuan unit item (sama dengan kolom Satuan di tabel inventaris).
7	Mutasi Terakhir	*Date*	Tanggal transaksi ini terjadi. Format: DD/MM/YYYY. Digunakan untuk menghitung interval antar pemakaian (TBF) dan fitur Days_Active.
8	Nomor WO	*String*	Nomor Work Order yang memicu transaksi pemakaian (mis. 01467/UM/UTS/2026). Bernilai "-" untuk transaksi non-pemakaian.
9	Work Center	*String*	Kode mesin atau area produksi yang membutuhkan item (mis. UTS-4008). Bernilai "-" jika tidak ada WO.
10	Keterangan	*String*	Jenis transaksi. Nilai: 1. pemakaian → stok keluar karena dipakai oleh mesin/WO. Dihitung sebagai transaksi KELUAR. 2. transfer barang keluar → stok dipindah dari gudang ini ke gudang lain. Dihitung sebagai transaksi KELUAR.

No	Nama Variabel	Tipe Data	Deskripsi & Nilai
			3. barang datang → item baru tiba dari supplier/pengadaan. Tidak dihitung sebagai pemakaian. 4. transfer barang datang → stok diterima dari gudang lain. Tidak dihitung sebagai pemakaian. 5. pengembalian → item dikembalikan ke gudang. Tidak dihitung sebagai pemakaian.
11	Asal Sloc	*Integer*	Kode gudang asal barang. Nilai umum: 1001 = gudang pusat/supplier, 1114/1115/1210/1211 = gudang per kategori, dll.
12	Sloc Tujuan	*String*	Kode gudang tujuan barang. Bernilai "-" jika barang keluar untuk pemakaian (bukan transfer). Berisi kode gudang jika transfer antar lokasi.
13	Nota Transaksi	*String*	Nomor nota/dokumen pendukung transaksi (mis. 0548/NSB/III/2026). Bernilai "-" jika tidak ada dokumen.
14	Sumber File	*String*	Path relatif file sumber dari mana baris ini digabungkan (mis. ../data/raw/report-mutasi-barang-elektrikA.xlsx). Hanya ada pada file report-mutasi-barang-gabungan.xlsx.

Tabel 3 3 Dataset Monitoring Pesanan (Purchase Order)

No	Nama Variabel	Tipe Data	Deskripsi & Nilai
1	NO	*Integer*	Nomor urut baris dalam laporan.

No	Nama Variabel	Tipe Data	Deskripsi & Nilai
2	STATUS	*String*	Status pemrosesan pesanan. Nilai: DRAFT → baru dibuat, PROSES PESANAN → sedang diproses internal, PROSES EVALUASI TEKNIS → evaluasi spesifikasi teknis, PROSES PO → purchase order sudah diterbitkan ke supplier, BARANG DITERIMA → item sudah tiba dan diterima, BARANG DITOLAK → item ditolak saat penerimaan, BATAL → pesanan dibatalkan.
3	TANGGAL	*Date*	Tanggal pesanan dibuat/dicatat. Format: DD-MM-YYYY.
4	KODE BARANG	*String*	ID unik item spare part. Kunci relasi ke tabel inventaris dan mutasi.
5	NAMA BARANG	*String*	Nama deskriptif item spare part yang dipesan.
6	WORK CENTER	*String*	Mesin atau area produksi yang membutuhkan item. Bernilai "-" jika tidak spesifik ke satu mesin.
7	JUMLAH	*Integer*	Kuantitas item yang dipesan (qty pesan).
8	SATUAN	*String*	Satuan unit item yang dipesan.
9	HARGA SATUAN	*Currency*	Harga per unit dalam Rupiah (IDR). Bisa bernilai 0 jika belum ada penawaran harga.
10	HARGA TOTAL	*Currency*	JUMLAH × HARGA SATUAN dalam Rupiah.
11	TANGGAL PO	*Date*	Tanggal Purchase Order resmi diterbitkan ke supplier. Bisa kosong jika belum sampai tahap PO.
12	JUMLAH TERIMA	*Integer*	Kuantitas aktual yang diterima dari supplier. Bisa berbeda dari JUMLAH jika ada kekurangan pengiriman. Bernilai kosong jika barang belum diterima.
13	ESTIMASI TIME DELIVERY (ETD)	*Date*	Tanggal estimasi barang akan tiba dari supplier. Format: DD-MM-YYYY.

No	Nama Variabel	Tipe Data	Deskripsi & Nilai
14	PERIODE (HARI)	*Integer*	Selisih hari antara TANGGAL PO dan tanggal barang diterima secara aktual. Bernilai negatif jika terjadi kesalahan input. Digunakan sebagai Lead_Time aktual per item. Nilai mean per Kode Barang → Lead_Time model. Nilai median seluruh PO (58 hari) digunakan sebagai default untuk item yang belum pernah dipesan.

2. Alat dan Bahan Penelitian

a. Perangkat Keras: Laptop dengan spesifikasi minimal Prosesor Intel Core i5, RAM 8 GB, dan penyimpanan SSD 256 GB.

b. Perangkat Lunak: Bahasa pemrograman Python untuk pemrosesan data dan pemodelan, *framework* Flask untuk pengembangan *backend* RESTful API, serta React.js (Vite) untuk pengembangan *frontend* website interaktif.

c. Pustaka (*Library*) Python yang digunakan dalam penelitian ini disajikan pada Tabel 3.2.

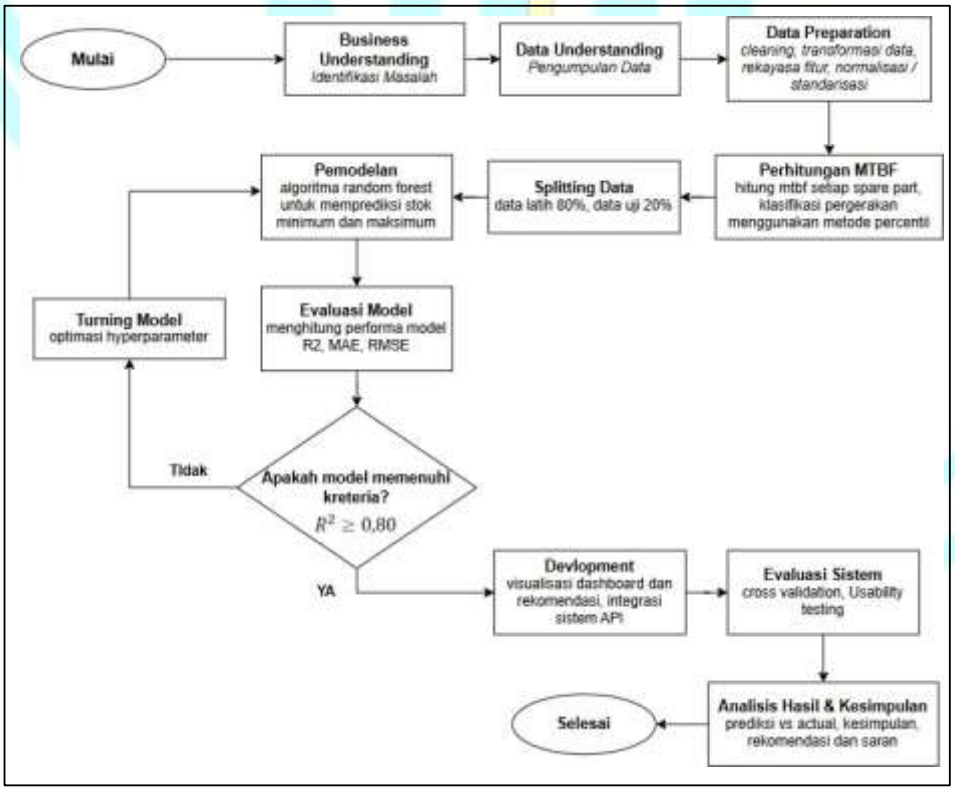
Tabel 3.4 Library Python yang Digunakan

Library	Modul	Fungsi
pandas, numpy	pd, np	Mengolah data transaksi, membersihkan data, dan manipulasi dataset
matplotlib.pyplot, seaborn	plt, sns	Membuat visualisasi data untuk EDA serta menampilkan grafik perbandingan metrik
sklearn.model_selection	Train_test_split	Membagi data menjadi data latih (*training set*) dan data uji (*testing set*)
sklearn.ensemble	RandomForestRegressor	Membangun model regresi menggunakan metode *ensemble Random Forest*
sklearn.metrics	r2_score, mean_absolute_error,	Menghitung metrik evaluasi regresi seperti R^2, MAE, dan RMSE

Library	Modul	Fungsi
	mean_squared_error	
flask	Flask()	Membangun Membangun RESTful API sebagai *backend* sistem prediksi
plotly	px, go	Membuat visualisasi data interaktif

3.2.2 Rancangan Penelitian

Penelitian ini mengadopsi kerangka kerja CRISP-DM (*Cross-Industry Standard Process for Data Mining*) dengan penyesuaian pada tahap akhir yang menambahkan evaluasi sistem dari perspektif pengguna. Kerangka ini dipilih karena sifatnya yang iteratif memungkinkan peneliti untuk kembali ke tahapan sebelumnya apabila ditemukan permasalahan baru di tengah jalan. Alur lengkap penelitian disajikan pada Gambar 3.1.



Gambar 3.1 Rancangan Penelitian

Berikut penjelasan masing-masing fase:

1. *Business Understanding*

   Langkah pertama adalah memahami akar permasalahan secara menyeluruh sebelum menyentuh data apapun. Di PT XYZ, dua masalah utama telah teridentifikasi: *stockout* yang menyebabkan mesin berhenti beroperasi karena komponen kritis tidak tersedia, dan *overstock* yang membuat modal kerja perusahaan tertahan pada stok yang tidak pernah terpakai. Dari pemahaman ini, tujuan penelitian dirumuskan dengan jelas: membangun sistem prediksi yang mampu mencegah kedua kondisi tersebut secara bersamaan.

2. *Data Understanding*

   Setelah permasalahan dipahami dengan baik, tahap berikutnya adalah mengenal data yang akan digunakan. Ketiga jenis data dikumpulkan dari sistem informasi internal perusahaan, lalu dieksplorasi untuk memahami karakteristiknya seberapa banyak item yang aktif bertransaksi, berapa yang berstatus diam (*non-moving*), serta bagaimana pola pemakaiannya berubah dari waktu ke waktu. Eksplorasi awal mencakup pemeriksaan tipe data, identifikasi nilai kosong pada kolom-kolom kritis, analisis distribusi jenis transaksi (keluar vs masuk), serta analisis struktur gudang untuk mendeteksi item yang tersimpan di lebih dari satu lokasi sekaligus. Temuan dari tahap ini menjadi dasar pengambilan keputusan teknis pada tahap Data Preparation berikutnya.

3. *Data Preparation*

   Tahap ini merupakan fondasi yang menentukan kualitas keseluruhan proses analisis, karena hasil pemodelan hanya akan sebaik data yang digunakan sebagai masukannya. Tahap ini terdiri dari empat sub-tahap teknis: pembersihan data, deteksi outlier, perhitungan metrik reliabilitas, dan rekayasa fitur (*feature engineering*).

   a. Pembersihan Data dan Standarisasi Format

      Data mentah dibersihkan dari nilai yang hilang, catatan duplikat, dan anomali. Format tanggal yang tidak seragam (mis. DD/MM/YYYY, YYYY-MM-DD, dengan/tanpa jam) distandarisasi ke satu format yang konsisten. Penulisan satuan yang bervariasi (pcs, PCS, PC) juga

diseragamkan ke format baku. Contoh transformasi format tanggal disajikan pada Tabel 3.3.

Tabel 3.5 Contoh Transformasi Format Tanggal

No	Data Awal	Setelah Transformasi	Keterangan
1	31/12/2024 08:30	31/12/2024	Format standar
2	15-Jan-24	15/01/2024	Format standar
3	NaT / kosong	Dihapus	Data tidak valid

b. Filtering Transaksi keluar dan Deteksi *Outlier* dengan IQR

Dari kolom Keterangan pada data mutasi, hanya baris dengan jenis transaksi ‘pemakaian’ dan ‘transfer barang keluar’ yang diambil sebagai dasar perhitungan, karena kedua jenis transaksi inilah yang merepresentasikan pengurangan stok riil. Selanjutnya, deteksi *outlier* dilakukan menggunakan metode *Interquartile Range* (IQR) dengan multiplier 3 untuk menyaring nilai-nilai ekstrem yang berpotensi mendistorsi statistik. Transaksi outlier tidak dihapus, melainkan ditandai (*flagged*) sebagai abnormal dan dikeluarkan dari perhitungan statistik agar dapat ditelusuri kembali apabila diperlukan. Contoh deteksi outlier disajikan pada Tabel 3.4.

Tabel 3.6 Contoh Deteksi Outlier dengan IQR

Item	Data Awal	Setelah Filtering	Catatan
SP001	5, 3, 4, 100	5, 3, 4	100 outlier (IQR upper: 8)
SP002	2, 3, 2, 4	2, 3, 2, 4	Tidak ada outlier
SP003	10, 8, 12, 500	10, 8, 12	500 outlier (IQR upper: 14)

c. Identifikasi Jenis Transaksi dan Pembuatan *Item Key*

Dari kolom Keterangan pada data mutasi, setiap transaksi dikategorikan sebagai KELUAR atau MASUK. Hanya dua jenis transaksi yang dikategorikan sebagai KELUAR dan digunakan sebagai dasar perhitungan: ‘pemakaian’ (stok keluar karena digunakan oleh mesin melalui *Work Order*) dan ‘transfer barang keluar’ (stok dipindahkan ke gudang lain). Tiga jenis transaksi lainnya dikategorikan sebagai MASUK dan tidak dihitung sebagai pemakaian, yaitu ‘barang datang’, ‘transfer barang datang’, dan ‘pengembalian’.

Selain itu, karena satu kode barang yang sama dapat tersimpan di lebih dari satu lokasi gudang (Sloc) dengan stok dan pola pemakaian yang berbeda, perlu dibentuk identifier unik yang disebut Item_Key. Item_Key dibentuk dengan menggabungkan kolom Kode Barang dan Asal Sloc menggunakan tanda pemisah “|”, misalnya “114-000022|1115” dan “114-000022|1211” masing-masing menjadi entitas yang berdiri sendiri. Pendekatan ini memastikan setiap kombinasi item dan lokasi gudang dianalisis secara terpisah, sehingga rekomendasi stok yang dihasilkan spesifik terhadap kondisi nyata di setiap gudang.

d. Perhitungan MTBF dan Klasifikasi *Movement*

Setelah data bersih, untuk setiap item dihitung nilai MTBF (*Mean Time Between Failures*) menggunakan Rumus (1) dan (2) yang telah dijelaskan pada Bab II. MTBF dihitung dari rata-rata interval waktu antara dua transaksi pemakaian berurutan untuk item yang sama. Selanjutnya, threshold klasifikasi movement ditentukan menggunakan pendekatan domain-based: Fast Moving (MTBF $\leq$ 60 hari), Medium Moving (60 < MTBF $\leq$ 180 hari), dan Slow Moving (MTBF > 180 hari). Threshold ini ditetapkan berdasarkan karakteristik operasional industri manufaktur dan siklus pemeliharaan mesin yang umum. Contoh perhitungan MTBF dan klasifikasi disajikan pada Tabel 3.5 dan Tabel 3.6.

Tabel 3.7 Contoh Perhitungan MTBF

Item	Tanggal Transaksi	TBF (hari)	MTBF
SP001	1 Jan, 15 Jan, 25 Jan, 10 Feb	14, 10, 16	13,3 hari
SP002	1 Jan, 1 Apr, 1 Jul	90, 91	90,5 hari
SP003	1 Jan, 1 Jul	181	181 hari

Tabel 3.8 Contoh Klasifikasi Movement

Item	MTBF	Pengecekan Threshold	Klasifikasi	Prioritas
SP001	13.3 hari	MTBF ≤ 60 hari	Fast Moving	Tinggi
SP002	90,5 hari	60 < MTBF ≤ 180 hari	Medium Moving	Sedang
SP003	200 hari	MTBF > 180 hari	Slow Moving	Rendah
SP004	-	Tidak ada transaksi	Non-Moving	Review

e. Perhitungan *Lead Time* dari Data *Purchase Order*

Lead time dihitung langsung dari kolom Periode (Hari) pada data *Purchase Order* menggunakan Rumus (4). Untuk setiap item, nilai *mean* dari seluruh historis pemesanan yang sudah selesai diambil sebagai Lead_Time aktual. Untuk item yang belum pernah dipesan sebelumnya, digunakan nilai median dari seluruh PO sebagai default sesuai Rumus (5).

f. Perhitungan *Safety Stock*, Min Stock, dan Max Stock

Setelah nilai MTBF dan Lead Time setiap item diketahui, langkah selanjutnya adalah menghitung parameter stok menggunakan formula Min-Max Stock. Pertama, dihitung Daily Usage sebagai estimasi pemakaian harian item dari total kuantitas yang keluar dibagi hari aktif. Kedua, dihitung Coefficient of Variation (CV) sebagai rasio standar deviasi terhadap rata-rata kuantitas pemakaian, yang mencerminkan

ketidakberaturan pola pemakaian item. Ketiga, Safety Stock dihitung menggunakan Rumus (7) dengan service level 95% (Z = 1,65). Keempat, Min Stock dan Max Stock dihitung menggunakan Rumus (8) dan (9). Terakhir, Order Quantity dihitung menggunakan Rumus (10) sebagai selisih antara Max Stock dan stok aktual saat ini. Status stok setiap item (Out of Stock, Below Minimum, Optimal, Above Maximum) ditentukan berdasarkan perbandingan antara stok aktual dengan batas Min dan Max yang telah dihitung.

g. Pemetaan Fitur Model ke Variabel Dataset Asli

Tahap akhir dari data preparation adalah membentuk 16 fitur input yang akan menjadi masukan model *Random Forest*. Setiap fitur tidak dibuat secara abstrak, melainkan diturunkan langsung dari kolom-kolom dataset asli yang telah dijabarkan pada Tabel 3.1. Pemetaan eksplisit antara fitur model dengan variabel dataset asli disajikan pada Tabel 3.7 informasi ini penting agar replikasi penelitian oleh peneliti lain dapat dilakukan secara akurat.

Tabel 3.9 Pemetaan Fitur Input Model ke Variabel Dataset Asli

No	Fitur Model	Asal Variabel Dataset	Cara Perhitungan
1	N_Trans	*Mutasi.Jumlah* (baris Pemakaian)	count(transaksi keluar) per item
2	Mean_Usage	*Mutasi.Jumlah* (baris Pemakaian)	mean(Jumlah) per item
3	Std_Usage	*Mutasi.Jumlah* (baris Pemakaian)	std(Jumlah) per item
4	Max_Usage	*Mutasi.Jumlah* (baris Pemakaian)	max(Jumlah) per item
5	Min_Usage	*Mutasi.Jumlah* (baris Pemakaian)	min(Jumlah) per item

No	Fitur Model	Asal Variabel Dataset	Cara Perhitungan
6	Total_Usage	*Mutasi.Jumlah* (baris Pemakaian)	sum(Jumlah) per item
7	Days_Active	*Mutasi.Mutasi Terakhir*	max(Tanggal) - min(Tanggal) per item, dalam hari
8	Trans_Frequency	derived: N_Trans / Days_Active	rasio jumlah transaksi terhadap rentang aktif
9	Usage_CV	derived: Std_Usage / Mean_Usage	Coefficient of Variation pemakaian
10	Usage_Range	derived: Max_Usage − Min_Usage	rentang variasi kuantitas pemakaian
11	Monthly_Std	*Mutasi.Jumlah* agregasi bulanan	std(sum_Jumlah_per_bulan) per item
12	Monthly_Max	*Mutasi.Jumlah* agregasi bulanan	max(sum_Jumlah_per_bulan) per item
13	Monthly_Trans_Avg	*Mutasi.Mutasi Terakhir* agregasi bulanan	mean(count_transaksi_per_bulan) per item
14	MTBF_Days	*Mutasi.Mutasi Terakhir*	rata-rata interval antar tanggal pemakaian (Rumus 1 & 2)
15	Failure_Rate	derived: 1 / MTBF_Days	kebalikan MTBF Rumus (3)

No	Fitur Model	Asal Variabel Dataset	Cara Perhitungan
16	Daily_Usage	*Mutasi.Jumlah* pada baris dengan *Keterangan = 'pemakaian'* atau *'transfer barang keluar'* (yaitu transaksi yang membuat stok berkurang)	Daily_Usage = Σ Jumlah Pemakaian ÷ Days_Active Merupakan estimasi pemakaian harian item, dihitung dari total kuantitas yang keluar selama periode aktif. Inilah variabel A yang dimaksud fitur yang asal-usulnya datang dari mutasi barang keluar.

4. *Modeling*

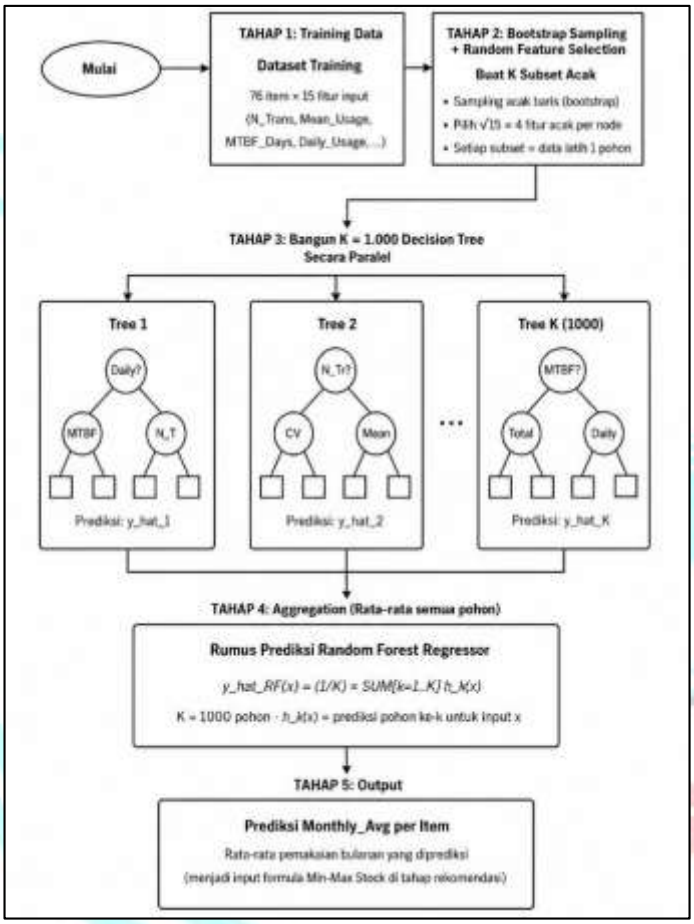
Tahap modeling adalah inti dari *pipeline* penelitian, di mana 16 fitur input yang telah dibentuk pada tahap Data Preparation digunakan untuk melatih algoritma *Random Forest Regressor* agar mampu memprediksi nilai Monthly_Avg (rata-rata pemakaian bulanan) per item. Nilai prediksi ini selanjutnya menjadi dasar perhitungan Min_Stock dan Max_Stock pada tahap rekomendasi.

a. Justifikasi Pemilihan *Random Forest*

Random Forest dipilih sebagai algoritma utama berdasarkan empat pertimbangan teknis yang relevan dengan karakteristik data *spare part.* Pertama, algoritma ini mampu menangkap pola *non-linier* pada data pemakaian yang tidak teratur kondisi yang umum ditemukan pada permintaan *spare part* yang dipengaruhi *breakdown* mendadak, jadwal perawatan berkala, dan faktor musiman. Kedua, *Random Forest* tahan terhadap *overfitting* berkat mekanisme bagging (*Bootstrap Aggregating*) dan *random feature selection*, sehingga prediksi yang dihasilkan lebih stabil dibandingkan *single decision tree*. Ketiga, algoritma ini tidak memerlukan normalisasi atau standardisasi data, yang menghemat waktu *pre-processing*. Keempat, *Random Forest* secara otomatis menghasilkan informasi *feature importance* yang berguna untuk interpretasi model dan validasi domain *knowledge.*

b. Prinsip Kerja *Random Forest*

Random Forest bekerja berdasarkan prinsip *ensemble learning*, yaitu menggabungkan prediksi dari sejumlah besar *Decision Tree* yang dilatih pada subset data dan subset fitur yang berbeda. Algoritma ini terdiri dari lima tahapan teknis yang dieksekusi secara berurutan, sebagaimana digambarkan pada Gambar 3.2.



Gambar 3.2 Tahapan Algoritma *Random Forest Regressor*

1) *Training* Data : Dataset *training* berisi item-item spare part yang memenuhi syarat kelengkapan data (rasio 80% dari total item yang valid) dengan 16 fitur input sebagaimana dijabarkan pada Tabel 3.9. Setiap baris merepresentasikan satu item dengan atribut statistik pemakaian, temporal, agregasi bulanan, dan reliabilitas.
2) *Bootstrap Sampling* dan *Random Feature Selection* : Dari dataset *training*, algoritma membentuk K subset acak menggunakan teknik *bootstrap*, yaitu pengambilan sampel acak dengan pengembalian

(*sampling with replacement*). Setiap subset memiliki ukuran yang sama dengan dataset asli namun dengan komposisi baris yang berbeda beberapa baris dapat terambil berkali-kali, sementara yang lain mungkin tidak terambil sama sekali (sekitar 36,8% baris yang tidak terambil disebut *out-of-bag* samples). Selain itu, pada setiap *node split*, hanya $\sqrt{16} \approx 4$ fitur yang dipilih secara acak dari 16 fitur yang tersedia (parameter max_features='sqrt'). Mekanisme ini disebut *random feature selection* dan berfungsi memastikan setiap pohon memiliki sudut pandang yang berbeda terhadap data, sehingga mengurangi korelasi antar pohon dan memperkuat efek *ensemble.*

3) Membangun K *Decision Tree* Secara Paralel : Sebanyak K = 1.000 *Decision Tree* dibangun secara independen, masing-masing dilatih pada subset *bootstrap*-nya sendiri dengan subset fitur acak yang berbeda di setiap node. Setiap pohon ditumbuhkan hingga kedalaman maksimum (parameter max_depth=None), artinya pohon dibiarkan tumbuh penuh hingga setiap *leaf node* memiliki sampel minimal (min_samples_leaf=1). Pada setiap node internal, algoritma memilih fitur dan *threshold* yang menghasilkan pengurangan variance terbesar pada target Monthly_Avg (kriteria *Mean Squared Error*). Karena setiap pohon dilatih pada data yang berbeda, masing-masing menghasilkan prediksi yang berbeda pula untuk input yang sama inilah dasar dari kekuatan *Random Forest*.
4) *Aggregation* (Rata-rata Prediksi) : Setelah seluruh 1.000 pohon selesai dibangun, prediksi akhir diperoleh dengan merata-ratakan output dari semua pohon untuk setiap input baru, sesuai Rumus (13) yang telah dijabarkan pada Bab II. Proses agregasi inilah yang disebut bagging gabungan dari banyak pohon yang *error*-nya saling meniadakan menghasilkan prediksi akhir yang jauh lebih stabil dibandingkan satu pohon tunggal. Secara matematis, jika pohon ke-k menghasilkan prediksi $h_k(x)$ untuk input x, maka prediksi Random Forest adalah $\hat{y}_{RF}(x) = (1/K) \times \Sigma\, h_k(x)$ untuk k = 1 hingga K.

5) Output Prediksi : Hasil akhir adalah nilai prediksi Monthly_Avg (rata-rata pemakaian bulanan) untuk setiap item *spare part*. Nilai inilah yang kemudian ditransformasi menjadi rekomendasi operasional melalui formula Min-Max Stock pada tahap berikutnya.

c. Pembagian Data dan Konfigurasi Hyperparameter

Data dibagi menggunakan fungsi train_test_split dari sklearn dengan rasio 80:20 (random_state=88), sehingga sebagian besar item digunakan untuk training dan sisanya untuk testing. Variabel target Monthly_Avg ditransformasi menggunakan log1p untuk menstabilkan varians dan mengurangi pengaruh skala yang besar antar item. Konfigurasi hyperparameter Random Forest yang digunakan dalam penelitian ini disajikan pada Tabel 3.10.

Tabel 3. 10 Konfigurasi Hyperparameter Random Forest

Hyperparameter	**Nilai**	**Justifikasi**
n_estimators	1000	Jumlah pohon yang dibangun. Nilai 1000 dipilih untuk menjamin kestabilan agregasi prediksi; semakin banyak pohon, semakin kecil variance prediksi (law of large numbers).
max_features	'sqrt'	Jumlah fitur yang dievaluasi pada setiap split = √(jumlah_fitur) = √16 ≈ 4. Nilai standar untuk task regresi yang menjamin keragaman antar pohon.
max_depth	None	Pohon dibiarkan tumbuh hingga kedalaman maksimum. Karena efek bagging, overfitting per pohon dapat ditoleransi karena akan dinetralisir oleh agregasi.
min_samples_split	2	Jumlah minimum sampel pada node internal untuk dapat dilakukan pemecahan (split). Nilai default sklearn.
min_samples_leaf	1	Jumlah minimum sampel pada leaf node. Nilai default — dipertahankan karena dataset relatif kecil sehingga setiap sampel berharga.
bootstrap	False	Karena ukuran dataset training relatif kecil, bootstrap di-disable agar setiap pohon

Hyperparameter	Nilai	Justifikasi
		menggunakan seluruh data training tanpa sampling ulang. Random feature selection tetap aktif untuk menjamin keragaman antar pohon.
random_state	42	Seed pseudo-random untuk reproducibility hasil eksperimen.

Apabila performa awal belum mencapai target ($R^2 \geq 0{,}8$), akan dilakukan *hyperparameter tuning* iteratif menggunakan grid search pada parameter n_estimators dan max_features hingga target performa tercapai.

5. Evaluasi Model

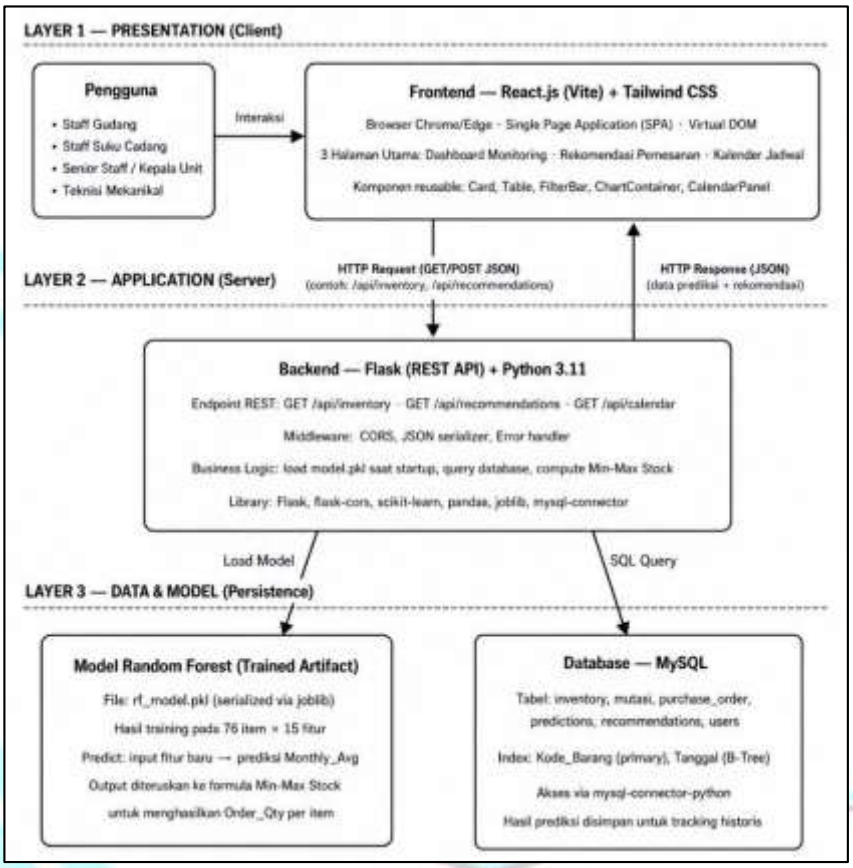
   Model yang telah dilatih dievaluasi menggunakan empat metrik secara bersamaan agar gambaran performa lebih menyeluruh: R^2 untuk mengukur seberapa baik model menjelaskan variasi data Rumus (17), MAE dan RMSE untuk mengukur magnitudo kesalahan dalam satuan asli (Rumus 14 dan 15), serta MAPE untuk menyatakan kesalahan dalam bentuk persentase Rumus (16). Selain itu, 5-fold *cross-validation* diterapkan untuk memastikan performa model konsisten di berbagai subset data dan tidak bergantung pada satu skema partisi tertentu.

   Setelah evaluasi metrik selesai, dilakukan analisis *feature importance* untuk mengidentifikasi fitur mana yang paling berpengaruh terhadap prediksi. Informasi ini berguna sebagai validasi *domain knowledge* sekaligus bahan pengembangan di penelitian selanjutnya. Hasil prediksi model kemudian diintegrasikan ke dalam formula Min-Max Stock untuk menghasilkan rekomendasi pemesanan akhir per item: Safety_Stock dihitung dengan Rumus (7), Min_Stock dengan Rumus (8), Max_Stock dengan Rumus (9), dan Order_Qty dengan Rumus (10).

6. *Deployment*

   Model *Random Forest* yang telah lolos tahap evaluasi tidak akan memberikan manfaat operasional apabila masih berbentuk *script* Python. Oleh karena itu, model perlu di-deploy ke dalam sebuah sistem berbasis web yang dapat diakses langsung oleh tim gudang dan pengadaan PT XYZ. Tahap

deployment dalam penelitian ini menggunakan arsitektur 3-tier yang memisahkan tampilan (*presentation*), logika aplikasi (*application*), dan penyimpanan data (*persistence*). Arsitektur sistem secara teknis disajikan pada Gambar 3.3.



Gambar 3.3 Arsitektur Deployment Sistem

Gambar 3.3 menggambarkan arsitektur sistem yang terbagi menjadi tiga layer dan dijelaskan tahapan teknisnya di bawah. Pendekatan 3-tier dipilih karena memberikan *separation of concerns* yang jelas: bagian tampilan tidak perlu tahu cara model bekerja, dan bagian model tidak perlu tahu bagaimana data ditampilkan.

a. *Layer Presentation (Frontend)* — React.js

   Layer presentation adalah lapisan yang langsung berinteraksi dengan pengguna. Lapisan ini dibangun menggunakan React.js dengan *build tool* Vite dan *styling* Tailwind CSS. Karena React menggunakan pendekatan *Single Page Application* (SPA), seluruh navigasi antar halaman terjadi tanpa reload penuh *Virtual* DOM hanya memperbarui bagian yang

berubah, sehingga tampilan terasa cepat dan responsif. *Frontend* terdiri dari tiga halaman utama (*Monitoring*, Rekomendasi Pemesanan, Kalender Jadwal) yang dibangun dari komponen *reusable* seperti *Card, Table, FilterBar, ChartContainer, dan CalendarPanel*. Komunikasi dengan server dilakukan melalui pemanggilan REST API menggunakan fetch atau axios.

b. *Layer Application (Backend)* — Flask REST API

Layer application berperan sebagai jembatan antara tampilan dan model *machine learning*. Lapisan ini dibangun dengan *framework* Flask di atas Python 3.11. Flask dipilih karena sifatnya yang minimalis (*micro-framework*) memberikan inti yang solid tanpa memaksakan struktur tertentu, sehingga komponen tambahan dapat dipilih sesuai kebutuhan. *Backend* menyediakan beberapa *endpoint* REST utama: GET /api/inventory untuk daftar stok, GET /api/recommendations untuk daftar rekomendasi pemesanan, dan GET /api/calendar untuk jadwal pemesanan. Setiap endpoint dilengkapi *middleware* untuk CORS (agar dapat diakses dari domain frontend), JSON *serializer* (untuk konversi DataFrame pandas ke JSON), dan *error handler*. *Library* yang digunakan meliputi Flask, flask-cors, scikit-learn, pandas, joblib (untuk loading model), dan mysql-connector-python (untuk akses database).

c. *Layer Persistence* (Model + Database)

Layer persistence menyimpan dua jenis aset: model *Random Forest* yang sudah dilatih dan basis data MySQL. Model di-*serialize* menggunakan joblib.dump() ke file rf_model.pkl yang berukuran kompak namun mempertahankan seluruh struktur 1.000 pohon keputusan. Pemilihan joblib dibandingkan pickle standar adalah karena joblib dioptimasi khusus untuk objek scikit-learn yang berisi array numpy berukuran besar. Sementara itu, MySQL digunakan untuk menyimpan tabel *inventory, mutasi, purchase_order, predictions, recommendations*. *Index B-Tree* ditambahkan pada kolom Kode_Barang dan Tanggal untuk mempercepat *query* yang paling sering dilakukan. Akses database dilakukan melalui mysql-connector-python yang sudah resmi didukung oleh Oracle.

d. Pengembangan *Backend* (Flask API)

Backend dibangun menggunakan file app.py sebagai titik masuk server Flask. Saat server pertama kali dijalankan, fungsi inisialisasi secara otomatis memanggil keempat skrip analisis secara berurutan: step0_eda.py untuk memuat dan menggabungkan data mentah, step1_data_processing.py untuk pembersihan dan perhitungan MTBF, step2_minmax_stock.py untuk komputasi *lead time* dan level stok, serta step3_ml_prediction.py untuk pelatihan dan prediksi model *Random Forest*. Seluruh hasil komputasi disimpan dalam memori server sebagai objek DataFrame global. Flask kemudian mendaftarkan tiga kelompok endpoint RESTful API utama: (1) GET /api/inventory untuk data stok dan rekomendasi pemesanan, (2) GET /api/summary untuk KPI ringkasan distribusi kategori *movement*, dan (3) GET /api/schedule untuk jadwal pemesanan terstruktur per tanggal. Semua respons diformat JSON dan dilengkapi header CORS menggunakan flask-cors.

e. Pengembangan *Frontend* (React.js)

Frontend dibangun menggunakan React.js 18 dengan build tool Vite dan diinisialisasi menggunakan perintah npm create vite@latest dengan template React. Struktur aplikasi menggunakan pendekatan component-based: setiap halaman (*Dashboard*, *Inventory*, *Calendar*) dikembangkan sebagai komponen React yang berdiri sendiri dan diatur oleh React Router untuk navigasi tanpa page reload. Setiap komponen memanggil endpoint API menggunakan hook useEffect, menyimpan data ke state menggunakan hook useState, lalu merender data secara dinamis. Logika filter berdasarkan kategori *movement*, Sloc, dan kata kunci pencarian diterapkan di sisi frontend menggunakan fungsi Array.filter(). Indikator warna status stok (merah, oranye, hijau, abu-abu) diimplementasikan menggunakan kondisi className Tailwind CSS yang dipilih secara dinamis. *Library* tambahan yang digunakan meliputi Recharts untuk visualisasi grafik, React-Calendar untuk tampilan kalender, dan Axios untuk komunikasi HTTP ke backend.

f. Tahapan Teknis *Deployment*

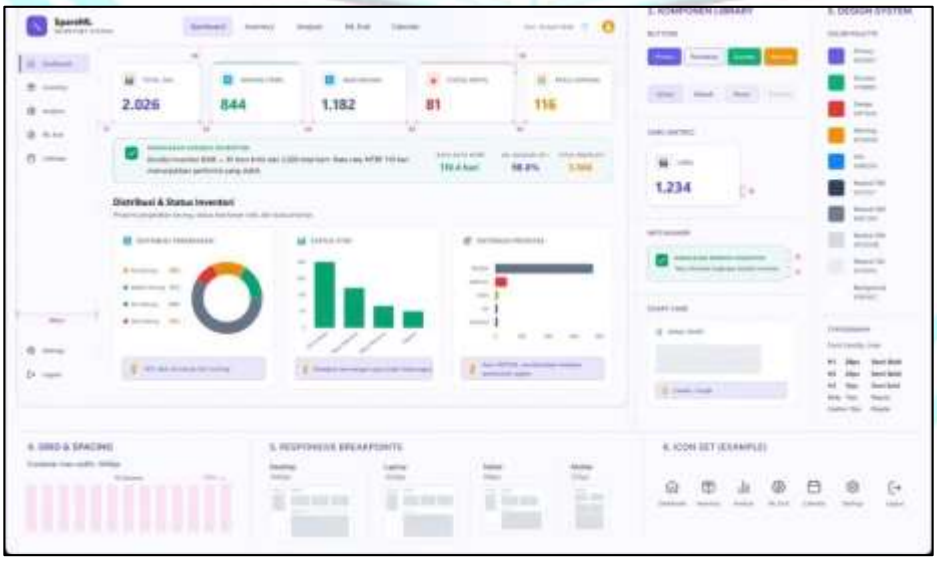
Tahapan deployment dilakukan secara berurutan dari proses preparation model hingga pengguna dapat mengakses sistem secara penuh, sebagaimana telah diilustrasikan pada bagian bawah Gambar 3.3.

1) Serialisasi Model : Model *Random Forest* yang sudah dilatih dengan konfigurasi pada Tabel 3.8 di-serialize ke file rf_model.pkl menggunakan perintah joblib.dump(model, 'rf_model.pkl'). File ini menyimpan seluruh struktur 1.000 pohon keputusan beserta parameter split di setiap node, sehingga model dapat di-load kembali tanpa perlu *re-training*.
2) *Loading* Model di *Backend* : Saat Flask app.py dijalankan pertama kali, model dimuat ke memori menggunakan joblib.load('rf_model.pkl'). *Loading* dilakukan satu kali di awal *startup*, kemudian objek model disimpan sebagai variabel global yang akan digunakan berulang kali oleh setiap *endpoint*. Pendekatan ini memangkas waktu inferensi karena *loading* model adalah operasi yang relatif mahal.
3) *Request* dari Pengguna : Pengguna mengakses sistem melalui browser dengan membuka URL aplikasi. React *frontend* kemudian mengirimkan HTTP GET request ke endpoint Flask sesuai halaman yang dibuka misalnya /api/recommendations saat halaman Rekomendasi Pemesanan dimuat. *Request* dikirim dalam bentuk JSON dengan parameter opsional seperti filter kategori *movement* atau pencarian kode barang.
4) Pemrosesan di *Backend* : Flask menerima *request*, lalu melakukan *query* ke MySQL untuk mengambil data *inventory* dan mutasi terbaru. Data tersebut diproses menggunakan pandas untuk membentuk 16 fitur input model (sesuai Tabel 3.7), kemudian fungsi model.predict(X_new) dipanggil untuk menghasilkan prediksi Monthly_Avg per item. Prediksi tersebut diteruskan ke modul perhitungan Min-Max Stock yang mengaplikasikan Rumus (6) hingga (10) di Bab II untuk menghitung Daily_Usage, Safety_Stock, Min_Stock, Max_Stock, dan Order_Qty.

5) *Response* ke *Frontend* **:** Hasil komputasi dirangkum dalam objek JSON yang berisi daftar item beserta rekomendasi pemesanannya, lalu dikirim balik ke *frontend* sebagai HTTP *Response*. React menerima JSON tersebut, melakukan *parsing*, lalu memperbarui state komponen sehingga tampilan otomatis ter-render ulang dengan data terbaru semua ini terjadi tanpa *reload* halaman berkat mekanisme Virtual DOM.
6) interaksi Pengguna : Pengguna melihat hasil dalam bentuk yang mudah dipahami: tabel item yang perlu dipesan dengan indikator warna prioritas, daftar tanggal pemesanan dalam kalender, dan ringkasan stok dalam *dashboard*. Pengguna dapat melakukan *filtering, sorting,* dan navigasi antar halaman tanpa perlu memuat ulang aplikasi. Apabila ada interaksi yang membutuhkan data baru (misalnya ganti filter), siklus kembali ke Tahap 3 secara transparan.

g. Rancangan Antarmuka Pengguna

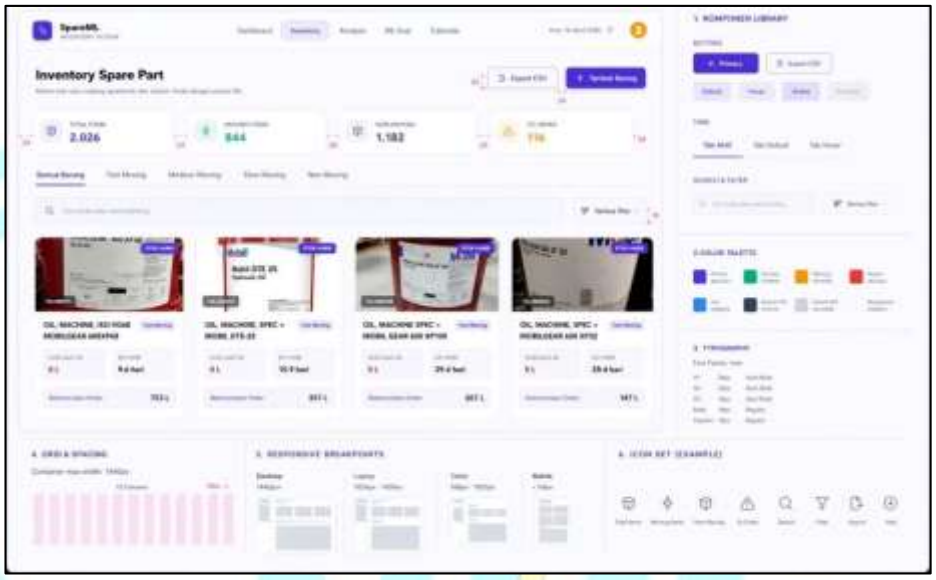
Sistem dirancang dengan filosofi yang mengutamakan kesederhanaan: tampilan yang bersih, informasi yang relevansinya langsung terasa, dan navigasi yang tidak memerlukan pelatihan khusus bagi pengguna awam sekalipun. Rancangan antarmuka tiga halaman utama tersebut disajikan pada Gambar 3.4, Gambar 3.5, dan Gambar 3.6 berikut.



Gambar 3.4 Rancangan Tampilan *Dashboard*

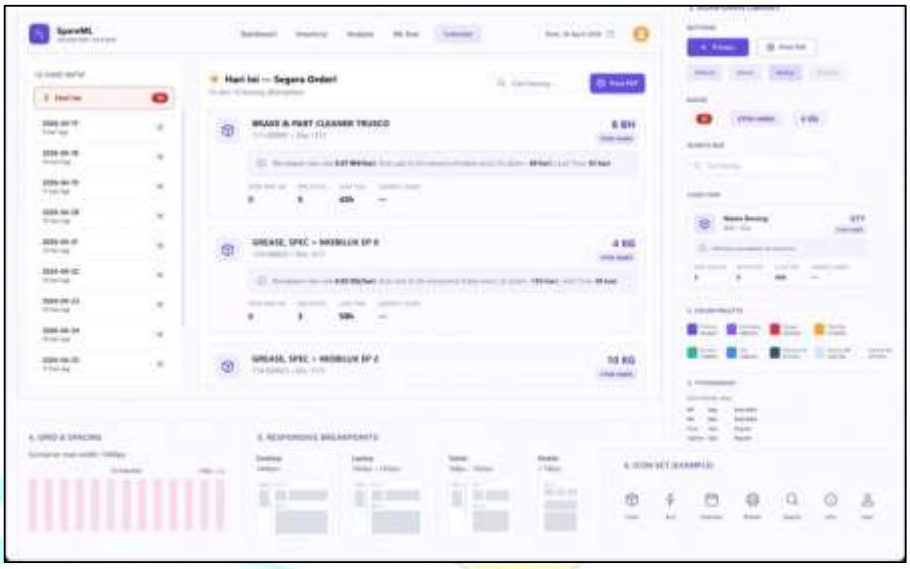
Gambar 3.4 menampilkan rancangan halaman *Dashboard* Utama yang menjadi pintu masuk pertama saat pengguna mengakses sistem. Dalam satu layar,

pengguna langsung disuguhkan empat kartu metrik ringkasan: Total SKU, *Moving Items*, *Non-Moving*, serta item yang berstatus kritis dan perlu segera dipesan. Di bawahnya, tiga grafik disajikan berdampingan untuk menggambarkan distribusi pergerakan barang, status stok saat ini, dan distribusi prioritas penanganan. Setiap grafik dilengkapi catatan insight yang memberi interpretasi otomatis agar pengguna tidak harus menafsirkan sendiri arti angka-angka tersebut.



Gambar 3.5 Rancangan Halaman *Inventory*

Gambar 3.5 memperlihatkan rancangan halaman *Inventory Spare Part*, yaitu tempat pengguna dapat menelusuri seluruh daftar suku cadang yang ada di gudang. Halaman ini dilengkapi tab navigasi untuk memfilter tampilan berdasarkan kategori pergerakan barang (Semua, *Fast Moving*, *Medium Moving*, *Slow Moving*, *Non Moving*), fitur pencarian berdasarkan kode atau nama barang, serta filter lokasi gudang. Setiap kartu item menampilkan foto barang, kategori pergerakannya, stok saat ini, estimasi MTBF, dan rekomendasi jumlah yang perlu dipesan. Pendekatan kartu bergambar dipilih karena lebih intuitif daripada tabel panjang yang sulit dibaca di lapangan.



Gambar 3.6 Rancangan Halaman Kalender Jadwal Pemesanan

Gambar 3.6 menampilkan rancangan halaman Kalender, fitur yang paling sering disebut oleh responden sebagai yang paling bermanfaat saat pengujian. Panel kiri menyajikan daftar tanggal-tanggal aktif berikut jumlah item yang perlu dipesan pada hari tersebut. Panel kanan menampilkan daftar item yang harus dipesan pada hari yang dipilih, lengkap dengan informasi kritis: stok saat ini, stok minimum, lead time, dan tingkat urgensi dalam satuan hari. Kombinasi tampilan kalender dan daftar detail seperti ini memungkinkan tim pengadaan merencanakan pembelian jauh lebih awal dan lebih terorganisir dibandingkan sistem berbasis laporan manual.

7. Evaluasi Sistem

   Sistem yang telah selesai dibangun diuji dari perspektif pengguna akhir melalui *Usability Testing* untuk menilai kemudahan dan kenyamanan penggunaan antarmuka. Instrumen yang dipakai adalah *System Usability Scale* (SUS), kuesioner standar internasional yang terdiri dari 10 pernyataan dengan skala Likert 1 sampai 4. Skor SUS dihitung menggunakan formula standar dan menghasilkan nilai dalam rentang 0–100. Sistem dianggap memiliki tingkat kegunaan yang baik apabila skor rata-rata berada di atas angka 68 atau masuk kategori *Good*. Daftar pertanyaan SUS yang digunakan disajikan pada Tabel 3.9.

Tabel 3.11 Pertanyaan SUS

No	Pertanyaan	Sangat Tidak Setuju	Tidak Setuju	Setuju	Sangat Setuju
		1	2	3	4
1	Saya akan sering menggunakan sistem ini untuk memudahkan pemesanan spare part	[ ]	[ ]	[ ]	[ ]
2	Sistem ini mudah dipahami dan tidak rumit	[ ]	[ ]	[ ]	[ ]
3	Fitur dan fungsi sistem terintegrasi dengan baik untuk proses pemesanan	[ ]	[ ]	[ ]	[ ]
4	Saya tidak memerlukan banyak dukungan teknis untuk menggunakan sistem ini	[ ]	[ ]	[ ]	[ ]
5	Rekomendasi pemesanan dari sistem akurat dan membantu mengambil keputusan	[ ]	[ ]	[ ]	[ ]
6	Sistem ini konsisten dan tidak ada ketidakkonsistenan dalam tampilan atau fungsi	[ ]	[ ]	[ ]	[ ]
7	Sebagian besar orang dapat dengan cepat belajar menggunakan sistem ini	[ ]	[ ]	[ ]	[ ]
8	Saya merasa percaya diri dan tidak bingung saat menggunakan sistem ini	[ ]	[ ]	[ ]	[ ]
9	Sistem ini dapat membantu menghindari kondisi stockout dan understock spare part	[ ]	[ ]	[ ]	[ ]
10	Saya percaya diri menggunakan sistem ini untuk mengelola persediaan spare part	[ ]	[ ]	[ ]	[ ]

Keterangan: STS = Sangat Tidak Setuju, TS = Tidak Setuju, S = Setuju, SS = Sangat Setuju.

8. Analisis Hasil & Kesimpulan

   Tahap terakhir dari kerangka CRISP-DM yang diadaptasi ini adalah melakukan analisis mendalam terhadap hasil yang diperoleh serta merumuskan kesimpulan. Pada fase ini, semua temuan dari berbagai tahapan penelitian dikompilasi dan dievaluasi untuk menentukan apakah tujuan penelitian telah tercapai. Analisis hasil mencakup: (1) perbandingan antara performa model prediksi dengan target awal ($R^2 \geq 0{,}8$ dengan MAE dan RMSE yang minimal); (2) validasi bahwa sistem dapat mengklasifikasikan suku cadang dengan tepat menggunakan MTBF; dan (3) verifikasi bahwa rekomendasi pemesanan otomatis yang dihasilkan sistem secara praktis dapat membantu mengurangi risiko *stockout* dan *overstock* di lingkungan operasional PT XYZ.