**SOFTWARE DEVELOPMENT PROJECT**

**PENGEMBANGAN WEBSITE PENJUALAN ONLINE DAN INVENTORI TOKO BANGUNAN DENGAN REACT.JS**



**Oleh:**

**Aaron Jason 223117070**

**Christophani G.P.W 223117073**

**Egbert Wangarry 223117080**

**Given Lee 223117082**

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS SAINS DAN TEKNOLOGI**

**INSTITUT SAINS DAN TEKNOLOGI TERPADU SURABAYA**

**SURABAYA**

**2025**

**SOFTWARE DEVELOPMENT PROJECT**

**PENGEMBANGAN WEBSITE PENJUALAN ONLINE DAN INVENTORI TOKO BANGUNAN DENGAN REACT.JS**

**Mengetahui/Menyetujui**

**Dosen Pembimbing**

**(Grace Levina Dewi, S.Kom., M.Kom.)**

**FAKULTAS SAINS DAN TEKNOLOGI**

**PROGRAM STUDI INFORMATIKA**

**INSTITUT SAINS DAN TEKNOLOGI TERPADU SURABAYA**

**SURABAYA**

**2025**

**KATA PENGANTAR**

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa, karena berkat rahmat, karunia, dan bimbingan-Nya, kami dapat menyusun dan menyelesaikan laporan “Pengembangan Sistem Informasi Manajemen Penjualan dan Inventori Berbasis Website untuk UD. Pelita Bangunan” dengan baik.

Proposal ini disusun sebagai bagian dari upaya mendokumentasikan proses perancangan dan pembangunan sistem yang diharapkan mampu memberikan solusi nyata terhadap permasalahan yang dihadapi perusahaan. Dalam era digital yang berkembang pesat, teknologi informasi telah menjadi salah satu faktor penentu keberhasilan bisnis. UD. Pelita Bangunan menyadari pentingnya penerapan sistem yang efisien, terintegrasi, dan adaptif untuk meningkatkan kualitas pelayanan kepada pelanggan sekaligus mendukung kelancaran proses kerja internal.

Laporan ini membahas berbagai tahapan pengembangan sistem, mulai dari analisis kebutuhan, perencanaan dan desain, implementasi, hingga pengujian serta evaluasi. Kami berharap laporan ini dapat memberikan gambaran yang jelas mengenai proses pengembangan serta manfaat yang dihasilkan, baik bagi peningkatan kualitas layanan maupun bagi perkembangan bisnis UD. Pelita Bangunan secara keseluruhan.

Ucapan terima kasih dan apresiasi kami sampaikan kepada seluruh anggota tim yang telah bekerja sama dengan penuh dedikasi dalam menyelesaikan laporan ini. Kami juga menyadari sepenuhnya bahwa laporan ini masih jauh dari sempurna. Oleh karena itu, segala bentuk kritik dan saran yang membangun akan sangat kami terima demi penyempurnaan di masa yang akan datang. Akhir kata, semoga laporan ini dapat memberikan manfaat dan menjadi kontribusi positif bagi kemajuan UD. Pelita Bangunan.

Surabaya, September 2025

Penulis

**DAFTAR ISI**

**Halaman**

HALAMAN JUDUL i

HALAMAN PENGESAHAN ii

KATA PENGANTAR iii

DAFTAR ISI iv

DAFTAR GAMBAR vi

DAFTAR TABEL viii

DAFTAR ALGORITMA ix

DAFTAR SEGMEN PROGRAM xi

BAB I PENDAHULUAN 1

1.1 Latar Belakang 1

1.2 Tujuan 2

1.3 Ruang Lingkup 3

1.4 Batasan 4

1.5 Uji Coba 5

1.6 Penjabwalan 5

1.7 Sistematika Pembahasan 7

BAB II ANALISA 10

2.1 Fact Finding 10

2.2 Deskripsi Sistem Lama 15

2.3 Analisa Masalah 17

2.4 Analisa Kebutuhan 18

2.5 Deskripsi Sistem Baru 20

2.6 Analisa Pembanding 23

2.7 Spesifikasi Kebutuhan 24

2.8 Test Case 27

2.8.1 Test Case Customer 27

2.8.2 Test Case Staff 29

2.8.3 Test Case Admin 29

2.9 Analisa Resiko 31

BAB III SCRUM I 33

3.1 Arsitektur Sistem 33

3.2 Product Backlog 34

3.3 Sprint Backlog 36

3.4 Pembuktian Sprint 39

3.5 Sprint Review 78

3.6 Burndown Chart 81

BAB IV SCRUM II 82

4.1 Product Backlog 82

4.2 Sprint Backlog 84

4.3 Pembuktian Sprint 86

4.4 Sprint Review 126

4.5 Burndown Chart 128

BAB V UJI COBA 129

5.1 Metode Pengujian 129

5.2 Skenario Pengujian Fungsional (Black Box) 129

BAB VI PENUTUP 139

6.1 Kesimpulan 139

6.2 Saran 140

LAMPIRAN A BURNDOWN CHART SPRINT 1 A-1

LAMPIRAN B BURNDOWN CHART SPRINT 2 B-1

LAMPIRAN C TABEL UJI COBA C-1

**DAFTAR GAMBAR**

**Gambar Halaman**

1.1 UD. Pelita Bangunan 2

1.2 Tabel Daftar Tugas 5

1.3 Gantt Chart Daftar Tugas 6

2.1 Bukti Wawancara kepada Pemilik UD. 11

2.2 Contoh Nota Pembelian Customer 14

2.3 Use Case Sistem Lama 16

2.4 Use Case Sistem Baru 20

2.5 Tampilan JuraganMaterial.id 23

3.1 Arsitektur Sistem 33

3.2 Bentuk Gambar Database 42

3.3 Gambar Database Login dan Register 46

3.4 Bentuk Gambar Login Register 47

3.5 Gambar Database Katalog dan Detail Produk 50

3.6 Bentuk Gambar Katalog Produk dan Detail Produk 51

3.7 Gambar Database Pencarian Barang 53

3.8 Bentuk Gambar Pencarian Barang 54

3.9 Gambar Database Menambah Produk dan Mengatur Produk di Cart 58

3.10 Bentuk Gambar Menambah Produk dan Mengatur Produk di Cart 59

3.11 Gambar Database Checkout dan Pembayaran 63

3.12 Bentuk Gambar Checkout dan Pembayaran 64

3.13 Gambar Database Status Pesanan dan Riwayat Transaksi 66

3.14 Bentuk Gambar Status Pesanan dan Riwayat Transaksi 66

3.15 Bentuk Gambar Halaman About Us 68

3.16 Bentuk Gambar Halaman FAQ 70

3.17 Gambar Database Fitur Notifikasi 72

3.18 Bentuk Gambar Fitur Notifikasi 72

3.19 Gambar Database Profile User 77

3.20 Bentuk Gambar Mengatur Profile User 78

3.21 Bukti Hasil Review Bersama Pemilik UD. 79

3.22 Grafik Burndown Chart Sprint 1 81

4.1 Gambar Database Stok Produk 88

4.2 Bentuk Gambar Mengelola Stok Produk 88

4.3 Gambar Database Pengembalian Produk 91

4.4 Bentuk Gambar Proses Permintaan Pengembalian Produk 91

4.5 Gambar Database Nomor Resi Pengiriman 93

4.6 Bentuk Gambar Menginput Nomor Resi Pengiriman 94

4.7 Gambar Database Menambah dan Mengelola Data Produk 99

4.8 Bentuk Gambar Menambah dan Mengelola Data Produk 100

4.9 Gambar Database Transaksi, Inventori dan Data Pengguna 104

4.10 Bentuk Gambar Laporan Transaksi, Inventori dan Data Pengguna 106

4.11 Gambar Database Voucher 110

4.12 Bentuk Gambar Menambah Voucher dan Mengelola Voucher 111

4.13 Gambar Database Akun Staff dan Admin 117

4.14 Bentuk Gambar Menambah, Mengelola, Mengaktifasikan Akun Staff 118

4.15 Gambar Database Kategori Produk 125

4.16 Bentuk Gambar Menambah, Mengelola Kategori Produk 126

4.17 Bukti Hasil Review Bersama Pemilik UD. 126

4.18 Grafik Burndown Chart Sprint 2 128

**DAFTAR TABEL**

**Tabel Halaman**

3.1 Tabel Product Backlog 34

3.2 Tabel Sprint Backlog 37

3.3 Tabel Hasil Sprint Review 79

4.1 Tabel Product Backlog 82

4.2 Tabel Sprint Backlog 85

4.3 Tabel Hasil Sprint Review 127

5.1 Tabel Uji Coba Navigation Bar 130

5.2 Tabel Uji Coba Kontak 132

5.3 Tabel Uji Coba Footer 133

**DAFTAR ALGORITMA**

**Algoritma Halaman**

3.1 Algoritma Database 39

3.2 Algoritma Login 43

3.3 Algoritma Register 44

3.4 Algoritma Katalog Produk 48

3.5 Algoritma Detail Produk 48

3.6 Algoritma Fitur Pencarian Barang 52

3.7 Algoritma Menambah Produk 55

3.8 Algoritma Mengatur Produk 56

3.9 Algoritma Checkout 60

3.10 Algoritma Pembayaran 60

3.11 Algoritma Melihat Status dan Riwayat Transaksi 65

3.12 Algoritma Melihat Halaman Informasi Toko 67

3.13 Algoritma Melihat Halaman Informasi Bantuan 69

3.14 Algoritma Melihat Notifikasi User 71

3.15 Algoritma Mengubah Profile User 73

3.16 Algoritma Mengubah Password User 74

3.17 Algoritma Menambah Alamat User 74

4.1 Algoritma Mengelola Stok Produk 87

4.2 Algoritma Permintaan Pengembalian Produk 89

4.3 Algoritma Menginput Nomor Resi Pengiriman 92

4.4 Algoritma Menambah Data Produk 94

4.5 Algoritma Mengelola Data Produk 96

4.6 Algoritma Lihat Laporan Transaksi 101

4.7 Algoritma Menambah Data Voucher 107

4.8 Algoritma Mengelola Data Voucher 108

4.9 Algoritma Menambah Akun Staff dan Admin 112

4.10 Algoritma Mengelola Akun Staff dan Admin 113

4.11 Algoritma Mengaktifasikan Akun Staff dan Admin 114

4.12 Algoritma Menambah Kategori Produk 119

4.13 Algoritma Mengelola Kategori Produk 120

4.14 Algoritma Mengelola Unit Konversi 121

**DAFTAR SEGMEN PROGRAM**

**Segmen Program Halaman**

3.1 Segmen Program Database 40

3.2 Segmen Program Login 44

3.3 Segmen Program Register 45

3.4 Segmen Program Katalog Produk 49

3.5 Segmen Program Detail Produk 49

3.6 Segmen Program Fitur Pencarian Barang 52

3.7 Segmen Program Menambah Produk 56

3.8 Segmen Program Mengatur Produk 57

3.9 Segmen Program Checkout 61

3.10 Segmen Program Pembayaran 62

3.11 Segmen Program Melihat Status dan Riwayat Transaksi 65

3.12 Segmen Program Melihat Halaman About Us 68

3.13 Segmen Program Melihat Halaman FAQ 70

3.14 Segmen Program Melihat Notifikasi User 71

3.15 Segmen Program Mengubah Profile User 75

3.16 Segmen Program Mengubah Password User 76

3.17 Segmen Program Menambah Alamat User 77

4.1 Segmen Program Mengelola Stok Produk 87

4.2 Segmen Program Permintaan Pengembalian Produk 90

4.3 Segmen Program Menginput Nomor Resi Pengiriman 92

4.4 Segmen Program Menambah Data Produk 97

4.5 Segmen Program Mengelola Data Produk 98

4.6 Segmen Program Lihat Laporan Transaksi 102

4.7 Segmen Program Menambah Data Voucher 109

4.8 Segmen Program Mengelola Data Voucher 109

4.9 Segmen Program Menambah Akun Staff dan Admin 115

4.10 Segmen Program Mengelola Akun Staff dan Admin 116

4.11 Segmen Program Mengaktifasikan Akun Staff dan Admin 116

4.12 Segmen Program Menambah Kategori Produk 122

4.13 Segmen Program Mengelola Kategori Produk 123

4.14 Segmen Program Mengelola Unit Konversi 124

**BAB I**

**PENDAHULUAN**

Proyek ini bertujuan untuk mengembangkan website sebagai langkah digitalisasi pada UD. Pelita Bangunan, sebuah usaha yang bergerak di bidang penjualan material bangunan. Dengan adanya website ini, proses penjualan dan pengelolaan inventori yang sebelumnya dilakukan secara manual dapat dilakukan secara lebih efisien, terintegrasi, dan real-time. Selain itu, sistem ini diharapkan mampu meningkatkan kualitas pelayanan kepada pelanggan serta mendukung perkembangan bisnis UD. Pelita Bangunan di era digital.

* 1. **Latar Belakang**

Toko Pelita Bangunan adalah toko bangunan yang menyediakan berbagai macam bahan untuk keperluan membangun dan merenovasi rumah. Di sini tersedia semen, cat, paku, kayu, besi, dan juga peralatan tukang. Salah satu produk terlarisnya adalah semen dan besi. Harganya cukup terjangkau dengan kualitas barang yang baik. Pelayanannya juga ramah sehingga membuat pelanggan merasa nyaman. Karena itu, banyak orang memilih Toko Pelita Bangunan sebagai tempat utama untuk belanja kebutuhan bangunan.

Operasional Toko Pelita Bangunan saat ini sepenuhnya bergantung pada sistem pencatatan manual, yang menimbulkan sejumlah kendala signifikan. Ketiadaan platform digital membuat jangkauan penjualan terbatas hanya pada pelanggan yang datang langsung, sekaligus menyebabkan data stok tidak dapat dipantau secara real-time. Lebih lanjut, proses manual ini tidak mampu mengakomodasi kompleksitas inventaris secara efektif, seperti konversi satu sak semen menjadi penjualan per kilogram, melayani permintaan berat kustom, serta mengelola pengembalian barang ke dalam stok. Akibatnya, muncul resiko tinggi terjadinya selisih stok, proses pemenuhan pesanan terhambat, dan analisis penjualan menjadi sulit dilakukan.

**Gambar 1.1**

**UD. Pelita Bangunan**

Untuk menjawab tantangan tersebut, dibangun sebuah Sistem Informasi Toko berbasis web. Sistem ini berfungsi sebagai platform digital untuk menampilkan katalog produk secara detail dan memfasilitasi transaksi penjualan secara online. Di sisi operasional, sistem ini dilengkapi fitur pengelolaan inventaris yang dirancang khusus untuk mengotomatisasi pemotongan stok dari konversi satuan, mengakomodasi penjualan kustom, serta menyediakan alur untuk proses retur termasuk validasi kondisi barang sebelum dikembalikan ke dalam pencatatan stok. Dengan demikian, implementasi sistem ini bertujuan memperluas jangkauan pasar, menjamin akurasi data inventaris, dan menghasilkan laporan penjualan yang terstruktur untuk mendukung pengambilan keputusan strategis.

* 1. **Tujuan**

Tujuan dari pembangunan sistem informasi berbasis website ini adalah untuk mendukung digitalisasi UD. Pelita Bangunan melalui penyediaan layanan yang terintegrasi, efisien, dan bermanfaat bagi pengelola maupun pelanggan. Secara khusus, tujuan yang ingin dicapai antara lain:

1. Membangun platform digital yang mampu menyajikan informasi produk meliputi deskripsi, harga, dan ketersediaan stok secara akurat dan real-time kepada pelanggan.
2. Meringankan beban kerja pengelola toko dengan mengotomatisasi proses pencatatan penjualan dan inventaris, sehingga dapat mengurangi risiko kesalahan data terutama dalam menangani konversi satuan, penjualan kustom, dan alur retur barang.
3. Mendukung pemilik toko untuk memperluas jangkauan pasar dan meningkatkan potensi pendapatan dengan memfasilitasi transaksi penjualan secara online.
4. Menyajikan ringkasan performa penjualan dan tren produk secara terstruktur untuk membekali pemilik toko dengan dasar pengambilan keputusan bisnis yang strategis.
   1. **Ruang Lingkup**

UD. Pelita Bangunan adalah toko bahan bangunan yang akan mengimplementasikan sistem informasi berbasis website untuk mendigitalisasi operasionalnya. Sistem ini dirancang untuk melayani berbagai stakeholder. Untuk pelanggan, sistem menyediakan pengalaman berbelanja online yang lengkap, memungkinkan mereka untuk menjelajahi katalog produk, melakukan transaksi pembelian, dan melacak riwayat pesanan mereka. Sementara itu, admin akan memiliki akses ke dashboard pengelolaan yang komprehensif untuk mengelola semua aspek internal toko.

Sistem ini akan dilengkapi dengan berbagai fitur inti. Bagi pelanggan, tersedia Fitur Katalog dan Transaksi yang mencakup fungsionalitas pencarian dan penelusuran produk, halaman detail produk, keranjang belanja, proses checkout terintegrasi dengan API RajaOngkir untuk estimasi ongkos kirim, riwayat transaksi, serta fitur live chat untuk komunikasi langsung dengan staff. Di sisi internal, Fitur Pengelolaan Toko (Dashboard Admin) akan memfasilitasi pengelolaan produk, sistem inventaris cerdas (termasuk penyesuaian stok otomatis, konversi satuan, penjualan kustom, dan retur), pemrosesan pesanan, serta pencatatan data supplier.

Sebagai penunjang keputusan bisnis, sistem juga dilengkapi dengan Fitur Laporan yang ekstensif. Laporan-laporan ini mencakup analisis penjualan (periodik, per kategori, berdasarkan metode pembayaran), laporan performa produk (terlaris, stok menipis, produk kurang laku), laporan retur barang, laporan pelanggan (pelanggan teratas, pendaftaran pelanggan baru), serta laporan ringkasan status pesanan. Fitur-fitur ini bertujuan untuk memberikan wawasan mendalam kepada admin , pemilik UD, staff, dan pelanggan. Pelita Bangunan dalam memantau kinerja, mengelola stok, dan merumuskan strategi bisnis yang lebih efektif.

* 1. **Batasan**

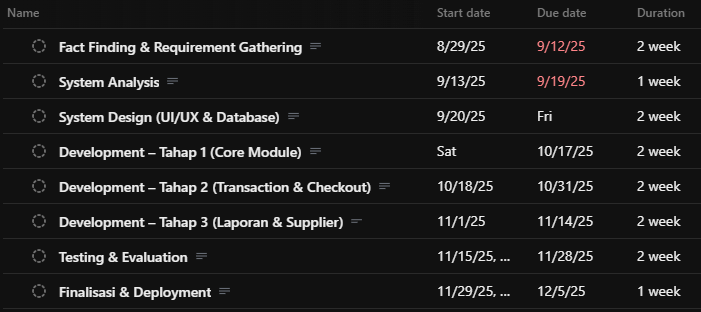
Pembuatan perangkat lunak tentu memiliki batasan yang harus diperhatikan. Hal ini berlaku juga pada perangkat lunak yang sedang dibuat. Di proyek Software Development Project kali ini memiliki beberapa batasan yaitu sebagai berikut:

1. **Fokus pada Transaksi Online:** Sistem ini hanya menangani pencatatan penjualan dan inventaris yang berasal dari transaksi online. Pencatatan transaksi yang terjadi secara langsung di toko fisik tidak termasuk dalam cakupan sistem.
2. **Integrasi Logistik Terbatas:** Fungsionalitas logistik terbatas pada integrasi dengan API RajaOngkir untuk menghitung estimasi biaya pengiriman secara otomatis saat proses checkout. Namun, cakupan sistem tidak termasuk fitur pelacakan paket (tracking) secara real-time atau pemesanan (booking) kurir. Oleh karena itu, admin tetap bertanggung jawab untuk memperbarui status pengiriman dan memasukkan nomor resi secara manual melalui dashboard.
3. **Pencatatan Data Supplier:** Fitur yang berkaitan dengan supplier terbatas pada fungsi pencatatan data. Sistem ini hanya digunakan untuk menyimpan informasi dasar seperti nama, kontak, dan alamat supplier untuk keperluan referensi internal. Sistem tidak menangani proses operasional seperti pembuatan pesanan pembelian (purchase order) atau integrasi dengan sistem pihak supplier.
   1. **Uji Coba**

Pengujian sistem akan difokuskan menggunakan metode Black Box Testing. Pada metode ini, pengujian dilakukan untuk memverifikasi fungsionalitas sistem dari sudut pandang pengguna, tanpa melihat struktur kode internalnya. Penguji akan fokus pada input (data yang dimasukkan) dan output (hasil yang diharapkan) untuk memastikan setiap fitur bekerja sesuai dengan spesifikasi kebutuhan. Jika terdapat output yang tidak sesuai, maka akan dicatat sebagai bug untuk diperbaiki pada fase implementasi.

* 1. **Penjabwalan**

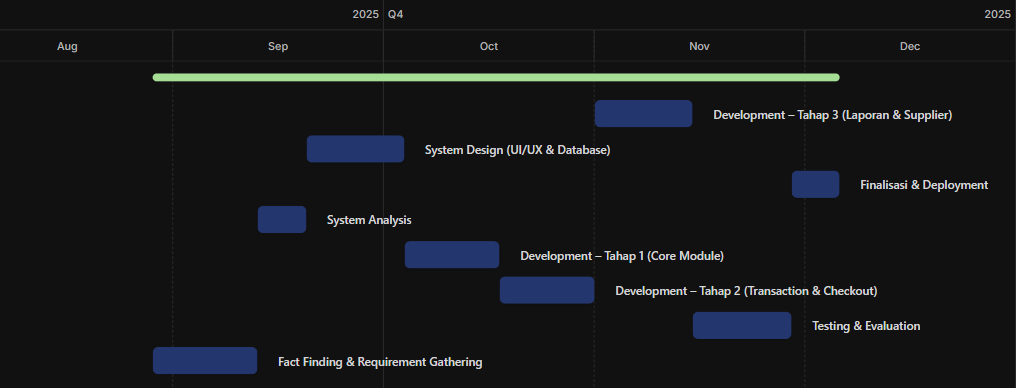
Pembuatan sebuah software tentu memerlukan waktu yang tidak cepat. Proses ini membutuhkan perencanaan yang matang agar setiap tahap pengerjaan proyek dapat berjalan lancar. Oleh sebab itu, dibutuhkan sebuah penjadwalan yang tersetruktur. Berikut adalah penjadwalan pengerjaan proyek dari awal hingga akhir:



**Gambar 1.2**

**Tabel Daftar Tugas**

Seperti yang ditunjukkan pada Gambar diatas, tabel tersebut menampilkan daftar tugas proyek beserta tanggal mulai (Start Date), tanggal selesai (Due Date), dan durasi waktu pengerjaan (Duration) untuk setiap tahap. Tahapan dimulai dari Fact Finding & Requirement Gathering hingga Finalisasi & Deployment, dengan total waktu pengerjaan sekitar tiga bulan. Setiap tahapan memiliki estimasi waktu yang berbeda sesuai tingkat kompleksitas dan ketergantungannya terhadap tahap sebelumnya.

****

**Gambar 1.3**

**Gantt Chart Daftar Tugas**

Selanjutnya, Gambar 1.7.2 memperlihatkan Gantt Chart dari daftar tugas tersebut. Gantt Chart ini menggambarkan alur waktu proyek secara visual, di mana setiap batang horizontal menunjukkan durasi pengerjaan masing-masing tugas. Dari bagan tersebut, dapat dilihat keterkaitan antar tahap—misalnya, System Analysis dimulai setelah tahap Fact Finding selesai, dan tahap Development terbagi menjadi tiga bagian yang berjalan berurutan. Visualisasi ini membantu tim proyek dalam mengawasi jadwal, memastikan setiap tugas selesai tepat waktu, dan menjaga alur kerja agar tetap efisien.

* 1. **Sistematika Pembahasan**

Pada buku ini, terdapat 6 bab yang terdiri dari Pendahuluan, Analisa, Desain, Implementasi, Uji Coba, hingga Penutup. Masing-masing bab menjelaskan tentang tiap proses pengerjaan proyek beserta hasilnya. Berikut adalah penjelasan detail mengenai setiap bab:

* BAB I : PENDAHULUAN

Pada Bab Pendahuluan, terdapat Latar Belakang yang menjelaskan alasan mengapa sistem informasi penjualan dan manajemen stok pada UD Pelita Bangunan perlu dibuat. Lalu, terdapat Tujuan yang menjelaskan sasaran dari pembuatan proyek ini. Selain itu, terdapat Ruang Lingkup yang menjelaskan cakupan sistem serta fitur-fitur utama yang akan dikembangkan. Bab ini juga memuat Batasan Masalah yang menjelaskan keterbatasan dari sistem, serta Sistematika Pembahasan yang menguraikan isi dari tiap bab laporan.

* BAB II : ANALISA

Pada Bab Analisa, terdapat Fact Finding yang menjelaskan metode pengumpulan data dari perusahaan. Selain itu, terdapat Deskripsi Sistem Lama yang menjelaskan proses bisnis manual yang masih berjalan. Lalu terdapat Analisa Masalah yang menguraikan kendala dari sistem lama, serta Analisa Kebutuhan yang memuat kebutuhan fungsional maupun non-fungsional sistem. Lalu terdapat Deskripsi Sistem Baru yang memberikan gambaran umum sistem yang akan dibangun, serta Spesifikasi Kebutuhan yang merinci fitur untuk masing-masing pengguna (admin, staff dan user). Lalu terdapat Test Case yang menunjukkan seluruh skenario pengujian menghasilkan output sesuai dengan hasil yang diharapkan. Selain itu, terdapat Analisa Resiko yang merupakan proses identifikasi, evaluasi, dan perencanaan strategi mitigasi untuk meminimalkan dampak negatif potensi masalah pada usaha.

* BAB III : SCRUM I

Pada Bab ini dijelaskan pelaksanaan siklus kerja Scrum pertama (Sprint 1) yang berfokus pada pembangunan arsitektur dasar sistem, termasuk backend, database, dan tampilan awal. Bab ini membahas proses perencanaan Sprint, pelaksanaan dan pelacakan menggunakan Daily Scrum dan Burndown Chart, serta Sprint Review yang menunjukkan hasil berupa sistem awal yang sudah fungsional dengan fitur autentikasi, pengelolaan produk, serta tampilan katalog dan pencarian produk.

* BAB IV : SCRUM II

Pada Bab ini dijelaskan pelaksanaan Scrum kedua (Sprint 2) yang berfokus pada penyempurnaan dan penambahan fitur lanjutan dari sistem sebelumnya. Tahap ini mencakup perencanaan, pelaksanaan, dan evaluasi hasil kerja tim dalam menyelesaikan fitur transaksi, manajemen pesanan, serta peningkatan antarmuka pengguna. Hasil dari Sprint 2 menghasilkan sistem yang lebih lengkap, stabil, dan siap untuk tahap pengujian dan implementasi berikutnya.

* BAB V : UJI COBA

Pada Bab Uji Coba, terdapat tabel-tabel hasil pengujian terhadap setiap fitur aplikasi menggunakan skenario yang telah dirancang. Hasil uji coba ini digunakan untuk memastikan sistem berjalan sesuai kebutuhan pengguna dan memperbaiki bug yang ditemukan.

* BAB VI : PENUTUP

Pada Bab Penutup, dijelaskan mengenai kesimpulan dari hasil pengembangan sistem, serta saran yang dapat dijadikan bahan pengembangan lebih lanjut di masa depan.

**BAB II**

**ANALISA**

Analisis merupakan tahap awal yang penting untuk memahami kebutuhan dan juga persyaratan pengguna untuk merancang sebuah sistem baru yang akan dikembangkan. Pengembang sistem dan pemangku kepentingan akan bekerja sama untuk mendefinisikan kebutuhan, fitur dan fungsi yang diperlukan agar sistem baru dapat memenuhi tujuan yang diinginkan. Hasil dari analisa ini akan menjadi dasar yang penting bagi fase pembangunan sistem selanjutnya.

**2.1 Fact Finding**

Pada sub-bab Fact Finding untuk pengembangan sistem penjualan dan inventori, langkah ini sangat penting untuk memahami secara mendalam kebutuhan dan harapan dari pengguna akhir maupun pemangku kepentingan lainnya. Fact Finding bertujuan untuk mengumpulkan informasi yang akurat mengenai sistem yang ada, masalah yang dihadapi, serta fitur yang diharapkan dari sistem yang akan dikembangkan. Teknik Fact Finding biasanya digunakan pada tahapan awal SDLC (System Development Life Cycle). Metode yang umum digunakan dalam tahap ini meliputi wawancara, observasi, dan analisis dokumen. Hasil dari proses ini akan menjadi pondasi penting bagi pengembangan solusi yang tepat serta memastikan bahwa produk akhir memenuhi tujuan bisnis yang telah ditetapkan.

**A. Wawancara**

Wawancara merupakan salah satu cara yang sering digunakan untuk mengumpulkan informasi atau data dari individu atau kelompok pemilik kepentingan. Wawancara dapat dilakukan oleh satu orang atau kelompok yang berperan sebagai pewawancara (interviewer). Proses wawancara ini tim pengembang lakukan secara tertulis kepada narasumber.

Berdasarkan hasil wawancara yang telah dilakukan dengan pemilik Usaha Dagang tersebut, dapat disimpulkan bahwa pemilik masih melakukan pengelolaan inventori dan penjualan secara manual, yaitu dengan pencatatan pada buku yang akan dilakukan oleh sang pemilik. Setelah itu pengaturan data seperti penjualan dari produk terlaris dan lain - lain bakal dilakukan secara manual juga dengan membaca buku penjualan. Selain itu, pengelolaan inventaris produk juga masih dilakukan secara manual dengan menghitung satu per satu produk yang tersisa dengan skala waktu perminggu. Sehingga waktu yang digunakan untuk mengolah data menjadi lama karena sang pemilik perlu melakukan kegiatan tersebut secara manual.

Berdasarkan hasil wawancara dengan pemilik usaha, dapat disimpulkan juga bahwa masalah yang ada di Usaha Dagang tersebut menyebabkan pengelolaan dan pemanfaatan data, seperti analisis data penjualan, menjadi kurang maksimal. Misalnya, ketika banyak customer dan juga suplier datang pada waktu yang sama akan membuat fokus menjadi terpecah sehingga menyebabkan pelayanan kepada customer menjadi kurang nyaman serta pemeriksaan produk menjadi kurang diperhatikan akibat kurangnya staff.  Selain itu, sang Pemilik belum menggunakan laporan secara periodik, sehingga ketika ingin melihat rekap data maka perlu mencari pada buku penjualan.



**Gambar 2.1**

**Bukti Wawancara kepada Pemilik UD.**

Pada bagian akhir, sang pemilik usaha menginginkan beberapa permintaan, yaitu agar pekerjaan pemilik bisa lebih dipermudah. Pada pembuatan sistem baru, akan lebih difokuskan pada bagian penjualan dan inventaris otomatis, serta ditambahkan pencatatan pembelian customer. Serta membuat tampilan website yang dibuat agar mudah dipahami dan nyaman digunakan.

**B. Observasi**

Observasi merupakan kegiatan yang dilakukan terhadap suatu proses atau objek dengan tujuan memahami fenomena berdasarkan pengetahuan dan gagasan yang telah ada. Kegiatan ini dilakukan untuk memperoleh informasi yang diperlukan dan dapat melengkapi hasil analisis yang didapatkan dari wawancara. Informasi yang didapatkan melalui observasi akan digunakan sebagai dasar untuk melanjutkan penelitian.

Berdasarkan hasil observasi yang dilakukan, terdapat beberapa informasi yang belum dijelaskan secara mendetail selama proses wawancara dilakukan. Sebagai contoh, dalam proses penerimaan barang dari supplier, adanya potensi ketidaksesuaian data stok produk. Sesuai dengan hasil wawancara, tim pengembang mengamati bahwa ketika pesanan produk tiba bersamaan dengan banyaknya customer, staff cenderung lebih memprioritaskan pelayanan customer. Akibatnya, produk yang baru datang seringkali langsung dimasukkan ke gudang penyimpanan tanpa melalui pengecekan yang mendetail. Kejadian tersebut meningkatkan resiko terjadinya ketidakcocokan antara stok produk dan juga pencatatan stok.

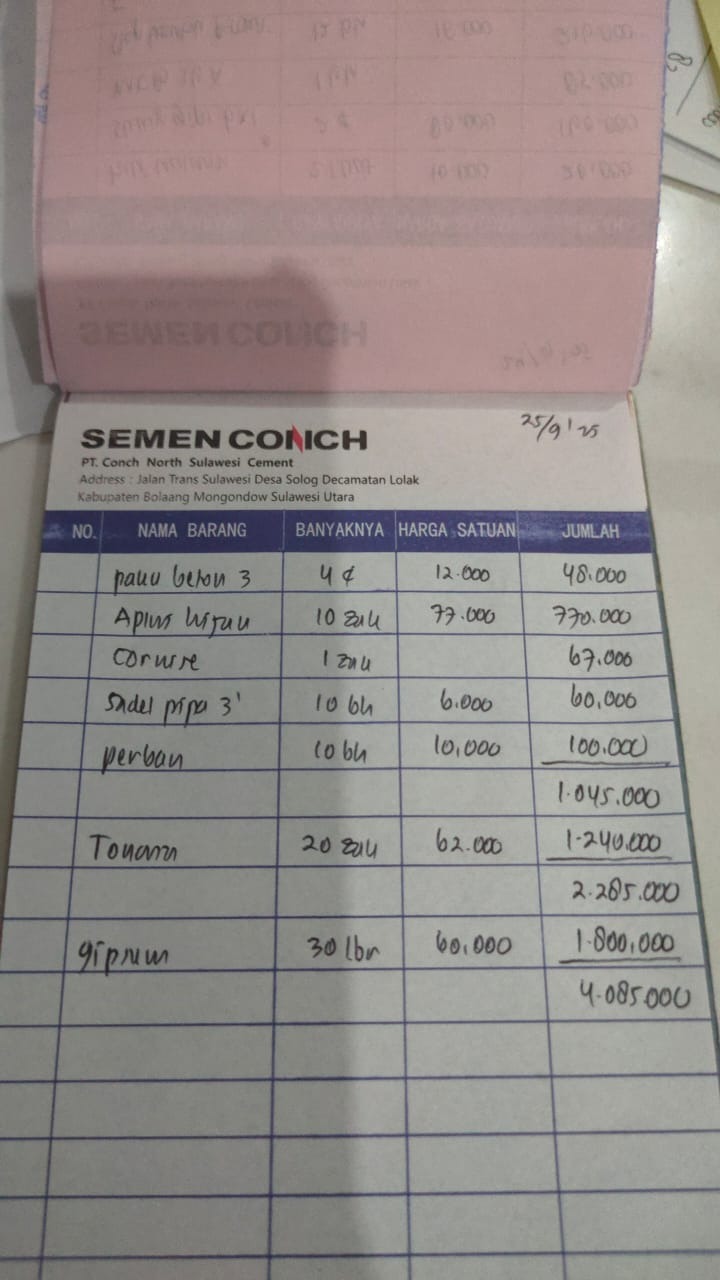
Observasi lebih lanjut pada proses penjualan dan pengembalian barang juga memberikan informasi tambahan. Tim pengembang mengamati langsung bagaimana staff melakukan pemeriksaan kondisi produk retur secara manual untuk memastikan jika produk sesuai dengan ketentuan yang ditetapkan oleh sang Pemilik, seperti produk yang dibeli merupakan cacat pabrik atau kesalahan pembeli, serta batas waktu pengembalian 1x24 jam. Selain itu, tim pengembang juga melihat kalau penjualan produk seperti semen, besi, sekrup, dan lain - lain tidak hanya dijual dalam satuan besar, tetapi bisa juga secara eceran (kustom), yang pencatatan pengurangan stoknya bergantung pada ingatan dan ketelitian staff saat mencatatnya di buku.

Terakhir, meskipun pada saat wawancara sang Pemilik merasa sistem manual saat ini masih bisa berjalan tanpa kendala yang berdampak besar pada penjualan toko saat ini, tentu masih ada kendala tersembunyi. Saat toko ramai, staff membutuhkan waktu ekstra untuk mencari sisa stok di gudang penyimpanan. Ketika ada barang yang habis, pencarian produk alternatif dengan fungsi serupa juga masih dilakukan secara manual. Tentu hal ini menjelaskan bahwa  meskipun sistem manual masih dapat dijalankan, prosesnya memakan waktu

**C. Analisa Dokumen**

Pada analisis dokumen, terdapat beberapa contoh yang memberikan gambaran lebih jelas mengenai pengolahan data pada sistem lama yang sedang digunakan. Beberapa dokumentasi, seperti formulir fisik, merupakan hasil dari analisis dokumen yang dilakukan. Dokumen - dokumen tersebut memberikan gambaran mengenai struktur pencatatan data yang diterapkan pada sistem lama Usaha Dagang. Contoh - contoh ini disajikan oleh pemilik sebagai ilustrasi umum dan bersifat gambaran kasar, namun tetap memberikan pemahaman yang cukup mendalam tentang bagaimana data dikelola sebelum adanya sistem yang lebih terstruktur.

Melalui analisis dokumen ini, dapat dilihat bahwa sistem lama masih menggunakan pencatatan manual, seperti penggunaan formulir fisik. Hal ini menunjukkan bahwa proses pengolahan data cenderung tidak terstandarisasi dan rentan terhadap kesalahan manusia, seperti produk yang dimasukkan tidak persis dengan yang dimiliki. Selain itu, dokumen yang ada juga memperlihatkan bahwa informasi sering kali terpisah tidak beraturan, sehingga menyulitkan proses pencarian dan pengelolaan data.



**Gambar 2.2**

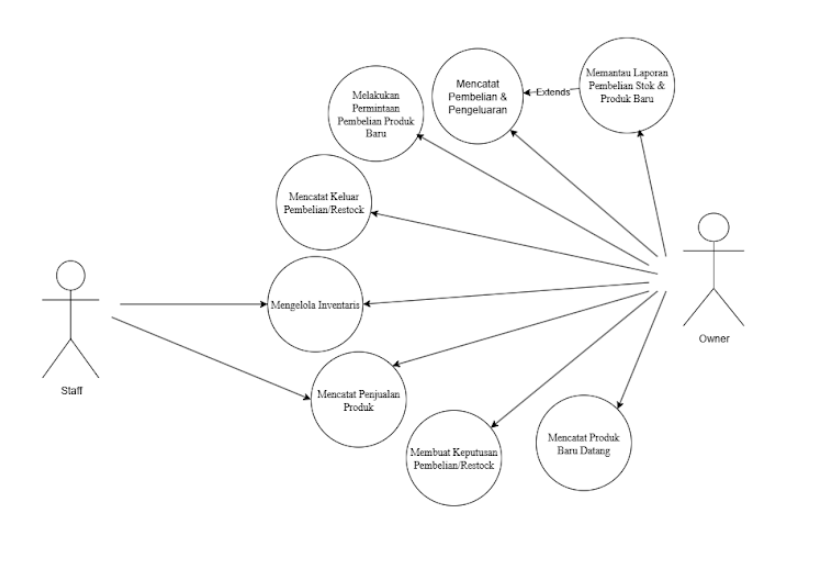
**Contoh Nota Pembelian Customer**

Pada gambar 2.1 memperlihatkan contoh tampilan nota pembelian yang diberikan kepada customer. Setelah nota ini diberikan kepada customer, produk yang terjual akan dicatat di buku pencatatan penjualan. Di buku pencatatan penjualan akan menyimpan data berupa tanggal berapa transaksi terjadi, produk apa yang terjual, berapa banyak stok yang terjual serta total harga dari penjualan.**2.2 Deskripsi Sistem Lama**

Saat ini, UD. Pelita Bangunan menjalankan proses penjualan secara tradisional, yaitu dengan pencatatan manual menggunakan buku. Metode tersebut diterapkan pada seluruh sistem, termasuk pada beberapa aspek utama, seperti inventaris yang pendataan produknya dilakukan satu per satu. Pada UD. tersebut, data yang paling sering dikelola adalah data mengenai stok produk dan juga penjualan, sehingga itu merupakan fokus utama dari pengelolaan data UD. tersebut.

Pada penggunaan sistem lama akan lebih mudah dipahami karena penggunaan sistem manual seperti pencatatan pada buku sudah sangat umum, sehingga mudah digunakan oleh banyak orang. Hal ini berlaku terutama pada staf yang kurang memahami teknologi yang ada. Selain itu, penggunakan sistem lama juga membuat UD. tidak perlu bergantung pada internet yang stabil dan juga teknologi yang canggih, sehingga tidak akan mengalami masalah seperti server yang tidak berfungsi dan berbagai masalah lainnya. Penggunaan sistem lama juga membuat staff tidak perlu khawatir dengan serangan cyber karena data tersimpan dalam penyimpanan lokal, seperti buku.  Dengan begitu, sistem pencatatan secara manual tidak perlu mengeluarkan biaya tambahan.

Kekurangan utama yang sering dijumpai pada sistem lama yaitu kurangnya ketepatan dalam pencatatan produk. Sebagai contoh, ketika suplier datang ke toko untuk ngerestok produk, staff perlu memperhatikan produk yang masuk secara teliti. Hal tersebut membuat staff perlu fokus agar tidak salah menghitung stok.

****

**Gambar 2.3**

**Use Case Sistem Lama**

Pengelolaan data pada sistem lama juga kurang baik dalam hal efisiensi waktu karena proses pencatatannya secara manual. Pencatatan dengan cara manual akan menghabiskan waktu yang cukup banyak, terlebih lagi jika perubahan data terjadi secara cepat. Sedangkan jumlah staf yang dapat memasukkan data juga terbatas.

Terdapat juga beberapa masalah lainnya yang terjadi karena menggunakan pengolahan data dengan sistem lama, yaitu tidak dapat diakses kapan saja dan dimana saja. Ketika pencatatan ditulis di buku, sang Pemilik tidak dapat mengelola data saat tidak memiliki buku tersebut. Kesulitan dalam pencarian data juga menjadi masalah karena sang Pemilik perlu membaca buku secara satu per satu, terlebih jika data yang tercatat sudah sangat banyak, sehingga mempersulit dan memperlama pekerjaan.

Kesulitan pencarian data juga akan menyebabkan masalah lain, yaitu memperlambat proses pengambilan keputusan. Hal ini terjadi karena seringkali data menjadi tidak terstruktur dengan baik sehingga memperlambat proses analisis dan pengambilan keputusan. Data pada sistem lama juga tidak dapat dipercaya sepenuhnya karena sulit melakukan pelacakan perubahan. Pengolahan sistem lama juga rentan terhadap kehilangan data jika terjadi bencana yang tidak diinginkan.

Sistem lama yang digunakan dalam pengelolaan data, terutama bagian penjualan dan inventaris, memiliki beberapa proses utama yang perlu diperhatikan. Setiap proses merupakan bagian penting pada pencatatan dan pengelolaan data pada sistem lama. Berikut ini adalah penjelasan lebih lanjut mengenai beberapa proses utama dalam sistem lama yang berkaitan dengan pengelolaan penjualan dan inventaris:

1. Pengelolaan Inventori

Pengelolaan Inventori biasanya dilakukan secara rutin, misalnya mengenai pencatatan keluar masuknya sebuah barang dan juga inventori akan diperbarui tiap minggu. Semuanya dilakukan secara manual dengan pencatatan di buku. Pengelolaan inventori bertujuan untuk mencatat masuk keluarnya stok produk.

**2.** Pembelian Produk Baru

Pembelian produk baru biasanya diawali dengan permintaan dari customer, selain itu bisa juga karena Pemilik ingin memperluas produk yang ingin dijual. Data produk yang ingin dibeli akan dicatat di buku. Jika produk tersebut telah dibeli, Pemilik akan memasukkan bukti pembelian ke dalam pendataan pembelian dan pengeluaran toko.

**2.3 Analisa Masalah**

Dalam menjalankan kegiatan usaha, ditemukan sejumlah hambatan yang timbul dari penggunaan sistem lama yang masih bersifat manual. Hasil wawancara dengan pemilik usaha menunjukkan bahwa proses pencatatan, pengelolaan, hingga analisis data belum berjalan dengan optimal. Permasalahan ini berdampak pada efisiensi waktu, akurasi data, serta kualitas pelayanan terhadap customer. Oleh karena itu, analisa masalah ini bertujuan untuk menguraikan kendala-kendala utama yang menjadi dasar perlunya pengembagan sistem baru.

1. Pada pengelolaan inventori yang masih dilakukan secara manual melalui buku catatan. Proses perhitungan stok barang dilakukan dengan menghitung satu per satu produk pada periode mingguan. Cara ini tidak hanya memakan waktu lama, tetapi juga beresiko tinggi menimbulkan kesalahan pencatatan serta membuat pemilik kesulitan memantau ketersediaan barang secara real-time.
2. Pencatatan transaksi penjualan pun masih dilakukan secara manual di buku penjualan. Akibatnya, analisis penjualan, seperti mengidentifikasi produk yang paling laku atau melihat trend penjualan, harus dilakukan dengan membaca ulang catatan satu per satu. Hal ini membuat proses analisis menjadi lambat, kurang efisien, dan menyulitkan pemilik ketika membutuhkan data secara cepat.
3. Hampir seluruh aktivitas usaha dikelola oleh pemilik tanpa adanya staff tambahan. Ketika jumlah customer dan melayani pemasuk datang secara bersamaan, pemilik seringkali kesulitan dalam membagi fokus antara melayani customer, mencatat transaksi, serta memeriksa inventory. Kondisi ini membuat pelayanan kepada customer menjadi kurang optimal dan rawan terjadi kesalahan dalam pengelolaan stok.
4. Usaha dagang ini juga belum memiliki sistem laporan berkala. ringkasan penjualan maupun inventori tidak tersedia secara otomatis, sehingga pemilik harus mencari kembali di buku catatan jika ingin mengetahui data tertentu. Proses tersebut memakan waktu cukup lama dan beresiko kehilangan informasi penting apabila catatan hilang atau rusak.

**2.4 Analisa Kebutuhan**

Berdasarkan permasalahan yang ditemukan pada sistem lama, diperlukan identifikasi kebutuhan yang dapat menjadi landasan dalam merancang sistem baru. Analisa kebutuhan ini berfungsi untuk menjabarkan fitur-fitur utama yang harus tersedia agar sistem dapat menyelesaikan kendala dan juga dapat meningkatkan kinerja usaha. Dengan merumuskan kebutuhan secara jelas, sistem yang dibangun diharapkan mampu menjawab permasalahan secara tepat sasaran serta memberikan manfaat nyata bagi pemilik usaha.

1. Dibutuhkan pengolahan data yang dilengkapi dengan pengecekan dan aturan saat mengelola data. Hal ini bertujuan untuk memastikan pencatatan stok maupun transaksi berjalan lebih akurat. Sebagai contoh, adanya pengecekan terhadap duplikasi data dan validasi jumlah stok yang diinput sesuai kondisi ril. Dengan demikian, kebutuhan ini dapat menjadi solusi atas permasalahan pencatatan manual yang sering menimbulkan kesalahan.
2. Dibutuhkan fitur pengelompokan data agar lebih terstruktur. Admin perlu memiliki kemampuan menambahkan kategori barang, jenis material, maupun data pemasok. Dengan adanya standar ini, proses pencatatan transaksi dan stok menjadi lebih mudah, konsisten, dan terarah. Kebutuhan ini menyelesaikan permasalahan pencatatan manual yang tidak memiliki format baku.
3. Dibutuhkan fitur laporan otomatis berdasarkan periode tertentu. Laporan ini dapat menampilkan informasi penjualan maupun inventori sesuai rentang waktu tertentu, seperti harian, mingguan, atau bulanan. Selain itu, laporan juga bisa memperlihatkan barang terlaris maupun stok yang menipis. Hal ini menjadi solusi dari permasalahan tidak tersedianya ringkasan penjualan yang cepat dan akurat.
4. Dibutuhkan fitur pengaturan hak akses untuk pengguna. Pemilik toko sebagai admin utama memiliki hak akses penuh, sedangkan staf hanya memiliki akses terbatas, seperti input transaksi atau memperbarui stok. Dengan adanya pembagian hak akses, beban kerja tidak menumpuk pada pemilik, keamanan data lebih terjaga, dan pekerjaan dapat lebih terdistribusi.
5. Dibutuhkan tampilan data yang selalu diperbarui secara langsung. Setiap transaksi penjualan atau perubahan stok akan langsung tercatat dalam basis data, sehingga informasi yang ditampilkan merupakan kondisi terbaru. Kebutuhan ini menjawab permasalahan kesulitan memantau ketersediaan barang secara real-time.

**2.5 Deksripsi Sistem Baru**

Pada sub-bab ini akan diberikan penjelasan mengenai deskripsi dari sistem baru yang akan dibuat. Sistem baru yang akan dibuat merupakan website yang akan digunakan oleh customer untuk membeli produk bahan bangunan, serta oleh staff dan Admin yang bekerja pada UD. Pelita Bangunan.

Customer dapat melakukan akses pada website untuk melihat produk bahan bangunan. Untuk membeli produk, customer wajib login ke akun terdaftar mereka. Jika belum memiliki akun, customer dapat melakukan pendaftaran terlebih dahulu pada website. Customer juga bisa melihat status pesanan yang sudah mereka checkout.



**Gambar 2.4**

**Use Case Sistem Baru**

Staff memiliki peran operasional yang lebih luas, yaitu dapat memproses pesanan yang telah dipesan oleh customer, serta menangani komplain, mengubah detail produk (seperti nama, harga, deskripsi, kategori), mengatur stok produk dan mengubah gambar produk.

Admin memiliki jangkauan kerja yang lebih luas dibandingkan dengan staff, dimana Admin dapat menambahkan produk baru, atau sekedar nge-restock ulang produk jualan, mengelola user yang terdaftar, serta melihat laporan - laporan yang ada.

Dengan demikian, hanya staff dan Admin yang akan bisa masuk atau login pada website yang akan dibuat (selain customer yang login untuk membeli). Ketika terdapat staff baru yang bekerja pada UD. Pelita Bangunan dan memerlukan akses pada website, maka Admin perusahaan perlu meregistrasikannya, sehingga staff tidak bisa melakukan registrasi sendiri pada website. Pemilik dari UD. Pelita Bangunan berperan sebagai Admin pada website ini, dan memiliki fitur spesial yaitu untuk menambahkan staff perusahaan baru yang akan menggunakan website tersebut.

Terdapat keunggulan yang dapat ditemukan dan dirasakan pada sistem baru ini. Seperti pengolahan data menjadi lebih terstruktur (terutama pada inventory bahan bangunan dan manajemen pesanan), dengan adanya sistem baru semua data akan tercatat dalam sistem database secara lebih baik, akan membantu pengolahan data. Proses pencatatan data pada sistem baru juga menjadi lebih terstruktur sehingga pencatatan data dapat lebih efisien.

Pada sistem baru, kita juga bisa melihat dan mengontrol data secara real-time dikarenakan semua data tercatat dan ter-update pada server secara langsung ketika terdapat perubahan data yang dimasukan oleh pengguna website tersebut. Dengan demikian, staff dan Admin dapat melihat dan memantau semua perkembangan data secara real-time dan dapat diakses di mana pun. Data juga akan menjadi lebih aman dikarenakan semua akan tercatat pada server, dan database dapat di-backup secara berkala. Sehingga lebih aman dari kerusakan fisik dan kehilangan data penting. Selain itu, kelebihan dari sistem baru adalah penyajian data menjadi lebih baik karena data dapat direpresentasikan dalam beberapa bentuk sebagai informasi yang bermanfaat untuk dianalisis, seperti laporan penjualan atau laporan stok. Dengan penggunaan sistem baru, standarisasi pada pencatatan akan menjadi lebih terkontrol, karena terdapat pengecekan yang lebih jelas, dan juga staff bisa melakukan pencatatan data, mengatur transaksi dan melayani customer.

Selain keunggulan yang ditawarkan oleh sistem baru, terdapat juga beberapa kekurangan, yaitu membutuhkan pemahaman teknologi. Dengan adanya sistem baru, maka pengguna perlu mempelajari cara menggunakannya terlebih dahulu. Selain itu, sistem baru akan bergantung terhadap internet karena sistem ini berbasis web, sehingga untuk mengakses website tersebut memerlukan sebuah perangkat yang terhubung dengan internet. Selain itu, akan lebih terancam oleh serangan siber, dikarenakan website sistem baru tersebut akan di-hosting, sehingga perlu lebih berhati-hati pada serangan siber. Kompatibilitas pengguna juga menjadi masalah, karena sistem baru berbasis teknologi, maka memerlukan kompatibilitas dengan perangkat yang digunakan pengguna (misalnya browser yang up to date). Modal yang diperlukan untuk menggunakan website tersebut juga akan lebih besar karena sistem ini akan memerlukan biaya, misalnya biaya hosting untuk bisa diakses secara daring dan biaya pengembangan sistem.

Pada sistem baru yang akan digunakan dalam pengolahan data, terutama bagian inventory dan penjualan, memiliki beberapa aktivitas atau proses utama yang perlu diperhatikan. Setiap aktivitas atau kegiatan tersebut akan memainkan peran penting pada pencatatan dan pengolahan data inventory dan penjualan. Berikut ini adalah penjelasan lebih lanjut mengenai beberapa proses utama dalam sistem baru yang berkaitan dengan pengelolaan inventory dan penjualan :

**1. Pembaruan dan Monitoring Stok Produk**

Pada penanganan data stok produk, seperti terdapat pembaruan data yang perlu dilakukan, hal tersebut akan lebih sering dilakukan oleh staff karena staff yang berhubungan langsung dengan pesanan customer. Admin juga dapat memperbarui data stok produk, tetapi staff akan lebih sering memperbarui status stok sisa produk. Sehingga Admin pada umumnya hanya akan melakukan monitoring terhadap data produk yang ada. Namun perlu diperhatikan bahwa tindakan yang dilakukan oleh staff tetap terbatas, seperti misalnya staff tidak bisa melakukan penambahan produk baru.

**2. Pembelian dan Pemrosesan Pesanan Customer**

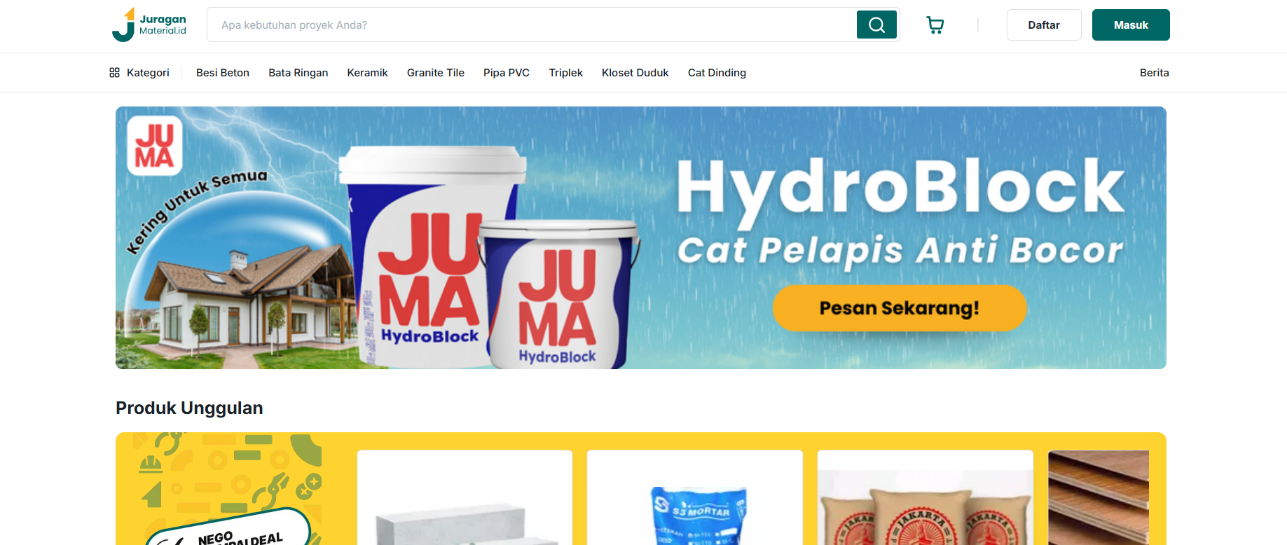
Customer jika ingin membeli barang maka perlu melakukan checkout. Sehingga staff akan menerima pesanan customer tersebut. Kemudian, staff akan memproses pesanan tersebut dapat berupa mempacking produk orderan customer. Jika pesanan sudah selesai di packing, maka hanya perlu untuk menunggu kurir untuk mengirim produk pesanan kepada customer. Data penjualan berupa bukti pembelian juga akan dimasukkan ke data penjualan.

**3. Melihat Laporan Keuangan dan Stok**

Pada sistem baru, Admin juga dapat melihat laporan yang lebih jelas mengenai data usaha dagang. Akan terdapat beberapa bagian yang dapat memberikan laporan yang hanya dapat dilihat oleh Admin. Seperti misalnya laporan inventaris (stok barang) dan laporan penjualan. Sehingga Admin bisa memilih laporan mana yang ingin dilihat, dan langsung saja menuju pada bagian yang diinginkan untuk melihat laporan tersebut.

**2.6 Analisa Pembanding**

Untuk memahami gambaran sistem yang akan dikembangkan, dilakukan analisis terhadap beberapa website mengenai penjualan bahan bangunan di Indonesia. Analisis ini bertujuan untuk mengidentifikasi model bisnis, dan fitur - fitur unggulan dari website sejenis yang bisa menjadi acuan ataupun pembeda untuk sistem yang akan kembangkan. Website yang terpilih sebagai objek perbandingan adalah ‘JuraganMaterial.id’.



**Gambar 2.5**

**Tampilan JuraganMaterial.id**

Website ini merupakan e-commerce berfokus pada penjualan ritel bahan bangunan secara lengkap dan terstruktur. Fitur utama dari website ini ada pada kelengkapan katalog produk yang luas, disertai dengan sistem navigasi dan pencarian yang efisien. Dari sudut pandang customer, proses transaksi mulai dari pemilihan produk, penambahan ke keranjang, hingga checkout dirancang agar mudah diikuti, mirip dengan marketplace pada umumnya. Model bisnis ini menunjukkan pentingnya memiliki struktur data produk yang solid dan alur transaksi yang lancar, yang menjadi salah satu acuan utama dalam pengembangan sistem untuk UD. Pelita Bangunan, terutama pada fitur katalog produk dan proses pemesanan oleh customer.

**2.7 Spesifikasi Kebutuhan**

Pada website UD. Pelita Bangunan ini, terdapat beberapa fitur yang dirancang untuk mengatasi permasalahan yang ada pada sistem lama serta memenuhi kebutuhan operasional. Website ini diharapkan dapat meningkatkan produktivitas dan kepuasan pengguna. Sistem baru ini akan memiliki beberapa jenis pengguna dengan peran yang berbeda-beda. Berikut adalah daftar fitur yang dikelompokkan berdasarkan peran masing-masing pengguna:

**1. Customer**

**a. Melihat Produk**

Customer dapat menelusuri katalog produk bahan bangunan yang tersedia di website.

1. **Mencari Produk**

Customer dapat mencari produk berdasarkan nama, kategori, atau filter lainnya.

1. **Melihat Detail Produk**

Customer dapat melihat informasi rinci (harga, deskripsi, gambar) dari setiap produk.

1. **Login Akun**

Customer wajib login ke dengan akun yang dibuat untuk dapat menambahkan produk ke keranjang belanja dan melanjutkan pembelian.

1. **Pendaftaran Akun**

Customer yang belum memiliki akun dapat register dan membuat akun baru pada website untuk dapat melakukan pembelian.

1. **Menambahkan Produk ke Keranjang**

Setelah login, customer dapat memasukkan produk yang diinginkan ke dalam keranjang belanja mereka.

1. **Melakukan Checkout**

Setelah login dan produk ada di keranjang, customer dapat melanjutkan ke proses checkout untuk menyelesaikan pembelian dengan mengisi detail pengiriman dan pembayaran.

1. **Melihat Riwayat Pesanan**

Customer yang sudah login dapat melihat riwayat semua pesanan yang pernah mereka lakukan.

1. **Melacak Status Pesanan**

Customer yang sudah login dapat melacak status pesanan mereka langsung dari akun.

**2. Admin**

**a. Login**

Admin perusahaan akan dapat melakukan login pada website. Dengan ketentuan yaitu kredensial login yang diberikan oleh Admin benar dan telah terdaftar pada sistem.

**b. Fitur Inventaris**

Admin akan dapat menambahkan barang baru pada inventaris perusahaan, selain itu memperbarui data inventaris (seperti jumlah stok, lokasi barang). Pada bagian inventaris, Admin akan paling sering melakukan pemantauan data seperti melihat riwayat perpindahan barang secara lebih jelas.

**c. Fitur Pembelian Produk Baru**

Admin akan dapat mencatat data pembelian, memperbarui data pembelian yang sudah ada pada perusahaan. Selain itu pada bagian ini Admin akan dapat melihat riwayat pembelian pada perusahaan.

**d. Fitur Pegawai**

Admin akan dapat menambahkan pegawai baru dengan informasi yang diperlukan, selain itu dapat memperbarui data pegawai tertentu dan menonaktifkan pegawai tertentu dari daftar aktif.

**e. Fitur Pengguna Sistem**

Admin dapat menambahkan pengguna yang dapat melakukan akses terhadap website. Sehingga ketika terdapat staff yang ingin menggunakan website, maka akun pengguna harus ditambahkan oleh Admin. Admin (pemilik perusahaan) juga akan memiliki peran khusus untuk dapat menambahkan Admin baru sebagai pengguna sistem.

**f. Melihat Laporan dan Pemantauan Data**

Admin akan dapat melihat laporan dan melakukan pemantauan data pada website tersebut. Laporan yang dapat dilihat seperti, laporan pengeluaran proyek, laporan perpindahan barang, dan laporan pembelian. Sedangkan pemantauan data berlaku pada keseluruhan data yang tercatat dan ditampilkan pada website.

**g. Melihat dan Mencatat Pengeluaran**

Admin akan dapat mencatat pengeluaran dan juga melihat semua pengeluaran yang dilakukan pada usaha dagang.

**3. Staff**

**a. Login**

Staff Pembelian perusahaan akan dapat melakukan login pada website. Dengan ketentuan yaitu kredensial login yang diberikan oleh Staf Pembelian benar dan telah terdaftar pada sistem.

**b. Menerima dan Memproses Pesanan**

Staf akan menerima data detail transaksi pembelian yang dilakukan oleh customer, memastikan kelengkapan data pembayaran, kemudian memproses (menyediakan) pesanan customer. Ini termasuk mencatat bukti pembayaran jika diperlukan.

**c. Melayani Pertanyaan Customer**

Staff dapat melihat informasi pesanan customer yang telah login, dan menjawab pertanyaan umum customer terkait produk atau status pesanan.

**d. Mengatur Pencatatan Data (Dasar)**

Staff akan dapat mencatat dasar yang dibutuhkan, seperti mengisi data pengiriman untuk pesanan customer, dan memasukkan informasi  awal untuk data inventaris sederhana.

**2.8 Test Case**

Tahap Test Case bertujuan untuk memastikan bahwa setiap fitur yang terdapat pada sistem baru berfungsi sesuai dengan kebutuhan dan spesifikasi yang telah dirancang. Uji coba dilakukan untuk memverifikasi keakuratan, konsistensi, dan keandalan sistem dalam menjalankan proses bisnis yang ada pada UD. Pelita Bangunan. Berikut adalah beberapa skenario pengujian utama yang dirancang berdasarkan fitur dari masing-masing pengguna sistem :

**2.8.1 Test Case Customer**

Pada saat customer ingin masuk ke dalam sistem, ia perlu menekan tombol “Login” di halaman utama. Setelah itu, customer akan diarahkan ke halaman login untuk mengisi username dan password miliknya. Jika data yang dimasukkan benar, maka sistem akan mengarahkan customer ke halaman utama. Namun, jika data yang dimasukkan salah, maka sistem akan menampilkan pesan error “Username atau Password salah.”. Jika customer belum memiliki akun, ia dapat menekan tombol “Register” untuk membuat akun baru terlebih dahulu.

Dalam proses pendaftaran akun, customer akan membuka halaman “Daftar” dan mengisi data identitas lengkap seperti nama, email, dan password. Setelah formulir pendaftaran terisi lengkap, customer perlu menekan tombol “Daftar” dan sistem akan menyimpan data akun baru ke dalam database. Jika berhasil, sistem menampilkan notifikasi “Registrasi Berhasil.” yang menandakan akun sudah aktif dan dapat digunakan untuk login.

Setelah berhasil login, customer dapat mengakses halaman “Produk” untuk melihat daftar produk yang tersedia. Sistem akan menampilkan katalog produk lengkap dengan nama, harga, kategori, dan stok yang masih tersedia. Customer dapat memilih kategori tertentu untuk mempermudah pencarian produk yang diinginkan.

Ketika customer menemukan produk yang ingin dibeli, ia dapat menekan tombol “Tambah ke Keranjang” pada produk tersebut. Sistem akan mencatat produk tersebut ke dalam daftar keranjang dan menampilkan jumlah serta total harga secara otomatis. Produk yang berhasil ditambahkan akan muncul di halaman keranjang belanja, di mana customer bisa mengubah jumlah pembelian atau menghapus item jika diperlukan.

Setelah semua produk yang diinginkan masuk ke keranjang, customer dapat melanjutkan proses checkout dengan menekan tombol “Checkout”. Pada tahap ini, customer akan diminta untuk mengisi data pengiriman seperti alamat dan memilih metode pembayaran. Setelah data lengkap dan konfirmasi dilakukan, sistem akan memproses transaksi dan menampilkan bukti pembelian yang dapat disimpan customer.

Untuk melihat perkembangan pesanannya, customer dapat membuka menu “Pesanan Saya” di halaman akun. Dengan memasukkan ID pesanan, sistem akan menampilkan status pesanan, apakah masih Diproses, sudah Dikirim, atau telah Selesai. Hal ini membantu customer memantau setiap tahapan pengiriman barang secara transparan.

**2.8.2 Test Case Staff**

Proses pertama yang dilakukan oleh staff adalah login ke sistem menggunakan username dan password yang telah diberikan oleh admin. Setelah menekan tombol “Login”, sistem akan memverifikasi kredensial. Jika valid, staff akan diarahkan ke halaman dashboard staff untuk mulai bekerja, namun jika data tidak cocok, sistem akan menampilkan pesan kesalahan login.

Ketika terdapat pesanan baru dari customer, staff dapat membuka halaman “Pesanan Masuk” untuk melihat daftar pesanan. Dengan memasukkan ID pesanan, sistem akan menampilkan rincian pesanan tersebut, termasuk nama customer, produk yang dibeli, dan status pesanan. Staff dapat memverifikasi pesanan dan melanjutkan prosesnya.

Setelah pesanan diterima, staff dapat memperbarui status pesanan menjadi “Dikirim” atau “Selesai” sesuai dengan proses yang sedang berlangsung. Staff hanya perlu memilih pesanan yang dimaksud, mengubah status, lalu menekan tombol “Simpan”. Sistem akan langsung memperbarui status pesanan pada akun customer secara real-time, sehingga customer dapat mengetahui perkembangan pesanannya.

Selain itu, staff juga memiliki akses untuk mengatur stok produk. Ketika stok barang bertambah atau berkurang, staff akan membuka menu “Stok Produk”, memilih produk berdasarkan ID produk, lalu memperbarui jumlah stok yang tersedia. Setelah disimpan, sistem akan memperbarui data pada database, sehingga informasi stok selalu akurat di halaman produk customer.

Selain itu, staff juga bertanggung jawab dalam mencatat data pengiriman. Staff akan membuka halaman “Form Pengiriman” dan mengisi informasi seperti nomor resi, nama kurir, dan alamat pengiriman. Setelah data dikirim, sistem akan menyimpannya ke dalam database dan menampilkannya pada riwayat pesanan, sehingga dapat diakses oleh customer dan admin kapan saja.

**2.8.3 Test Case Admin**

Sebagai pengguna dengan akses tertinggi, admin akan memulai dengan melakukan login ke sistem menggunakan username dan password admin. Setelah menekan tombol “Login”, sistem akan memvalidasi data tersebut. Jika benar, admin diarahkan ke halaman dashboard utama yang berisi ringkasan aktivitas toko. Jika salah, sistem akan menampilkan pesan error dan meminta admin untuk mencoba kembali.

Admin memiliki kewenangan untuk menambahkan produk baru ke dalam katalog. Caranya dengan membuka menu “Tambah Produk”, lalu mengisi form dengan data lengkap seperti nama produk, harga, kategori, stok, dan deskripsi produk. Setelah menekan tombol “Simpan”, sistem akan menyimpan data tersebut dan menampilkan produk baru di halaman katalog yang dapat dilihat oleh customer.

Selain itu, admin juga dapat mengelola akun staff, baik untuk menambah staff baru maupun menonaktifkan akun lama. Pada halaman “Manajemen Staff”, admin dapat memasukkan nama, username, dan role staff, kemudian menekan tombol “Tambah”. Data staff yang baru akan tersimpan dan muncul di daftar pengguna aktif, sedangkan akun yang dihapus akan langsung dinonaktifkan dari sistem.

Admin juga dapat mengakses laporan penjualan dengan memilih menu “Laporan Penjualan”. Di sana, admin dapat menentukan rentang tanggal atau periode laporan, lalu sistem akan menampilkan data transaksi penjualan yang terjadi selama periode tersebut. Fitur ini memudahkan admin dalam menganalisis performa penjualan toko.

Selain penjualan, admin dapat mencatat pengeluaran toko, seperti biaya operasional atau pembelian stok barang. Admin membuka form “Pengeluaran”, mengisi jenis pengeluaran dan nominalnya, lalu menyimpan data tersebut. Setelah itu, sistem akan menampilkan pengeluaran tersebut pada laporan keuangan agar dapat dipantau secara berkala.

Terakhir, admin dapat menggunakan fitur Monitoring Stok dan Pembelian untuk memantau kondisi inventaris secara menyeluruh. Dengan memasukkan ID produk atau transaksi, admin dapat melihat data stok terkini serta riwayat pembelian yang tersimpan di sistem. Fitur ini membantu admin memastikan bahwa ketersediaan barang selalu terjaga dan aktivitas toko berjalan efisien.

**2.9 Analisa Resiko**

Analisa resiko merupakan proses mengidentifikasi, mengevaluasi, dan merencanakan tindakan apa yang akan diambil terhadap potensi masalah yang dapat menghambat atau mengancam keberhasilan proyek. Tujuan dari analisis ini adalah untuk meminimalkan dampak negatif  dari resiko - resiko tersebut dengan mempersiapkan strategi mitigasi yang tepat. Berikut adalah resiko yang dapat diperoleh beserta cara mengatasi resiko tersebut :

**1. Resiko Teknis**

Resiko teknis berhubungan dengan teknologi dan infrastruktur yang digunakan dalam sistem ini. Resiko pada kategori ini dapat berupa potensi masalah keamanan siber dan juga ketergantungan pada layanan internet yang stabil, yang merupakan kelemahan pada sistem baru yang berbasis web.

* Resiko keamanan siber dapat dicegah dengan upaya menggunakan enkripsi data untuk informasi sensitif, melakukan backup data secara berkala, serta menerapkan sistem autentikasi yang kuat.
* Resiko ketergantungan pada layanan internet dapat di cegah dengan upaya memilih layanan yang memiliki reputasi dan jaminan yang baik, serta menyediakan panduan bagi pengguna tentang apa yang harus dilakukan jika sistem sedang offline.

**2. Resiko Adaptasi Pengguna**

Resiko ini berpusat pada faktor manusia, mengenai bagaimana pengguna akhir menerima dan berinteraksi dengan sistem baru. Salah satu faktor keberhasilan sistem ini juga bergantung pada kesiapan pengguna akhir dalam menerima sistem baru.

* Resiko penerimaan perubahan dapat di tangani dengan cara melibatkan pengguna secara aktif selama proses pengembangan sistem ini, memastikan interface dapat mudah dipahami, serta menyediakan dokumentasi panduan yang terstruktur.
* Resiko kesalahan input (Human Error) dapat ditangani dengan cara menerapkan validasi input pada setiap formulir untuk meminimalkan kesalahan.

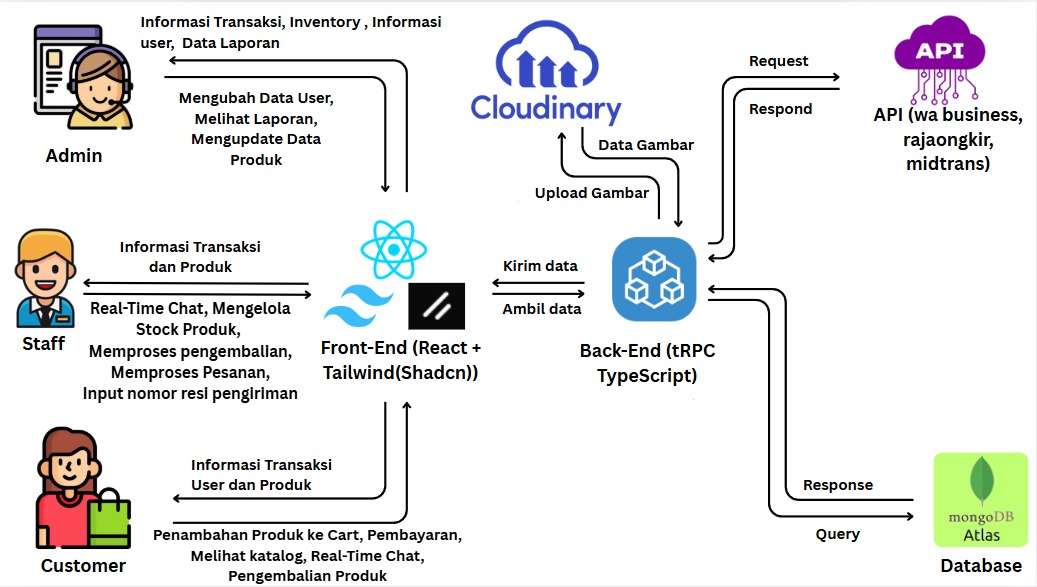
**BAB III**

**SCRUM I**

Pada bab ini, menjelaskan mengenai pelaksanaan siklus kerja Scrum pertama, atau Sprint 1. Fokus utama dari Sprint 1 adalah untuk membangun arsitektur dasar sistem serta mengimplementasikan fitur-fitur yang memungkinkan produk dapat diuji dan divalidasi pada tahap paling awal. Sprint ini meletakkan fondasi bagi pengembangan fitur-fitur yang lebih kompleks pada sprint berikutnya.

**3.1 Arsitektur Sistem**

Sebagai landasan untuk mencapai Sprint Goal, langkah pertama yang perlu dilakukan adalah merancang arsitektur sistem secara keseluruhan. Arsitektur ini berfungsi sebagai blueprint untuk tim pengembang dalam membangun sistem dan memastikan semua bagian dapat terintegrasi dengan baik. Gambar dibawah ini akan memberikan gambaran besar mengenai arsitektur sistem yang akan dibuat.



**Gambar 3.1**

**Arsitektur Sistem**

Berdasarkan diagram diatas, bagian frontend akan menggunakan React + Tailwind + Shadcnuntuk antarmuka yang akan berinteraksi langsung dengan pengguna. Kemudian menggunakan tRPC TypeScript untuk backend dari sistem yang dibuat, dimana backend akan menerima permintaan dari frontend, memproses logika sistem, serta berhubungan langsung dengan database. Untuk database, tim pengembang menggunakan MongoDB Atlasuntuk menyimpan semua data sistem, seperti data pengguna, data produk, serta data transaksi. Dan terakhir tim pengembang menggunakan Layanan Eksternal (API) agar sistem terhubung dengan layanan pihak ketiga melalui API, seperti WA Business untuk fitur chat, RajaOngkir dan Midtrans untuk fungsionalitas pengiriman dan pembayaran di masa mendatang.

**3.2 Product Backlog**

Produk Backlog adalah daftar utama yang berisi semua kebutuhan dan fitur yang inginkan untuk system. Dimana berdasarkan system yang akan kembangkan, tim pengembang dapat memilah fitur - fitur apa saja yang memiliki prioritas lebih besar dibanding fitur lainnya. Tabel dibawah ini akan memberikan penjelasan lebih jelas mengenai fitur - fitur apa saja yang dipilih untuk dikerjakan :

**Tabel 3.1**

**Tabel Product Backlog**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Deskripsi** | **Aktor** | **Prioritas** | **Sprint** | **Waktu** |
| PB - 1 | Masukan Data ke Database | Admin | 1 | 1 | 6 Jam |
| PB - 2 | Mendaftar serta login ke akun yang dibuat agar bisa mengakses fitur pembelian dan melacak pesanan. | Customer | 1 | 1 | 26 Jam |
| PB - 3 | Melihat katalog produk lengkap dengan detail dari barang yang dipilih agarmendapat informasi lebih mengenai produk yang ingin dibeli. | Customer | 2 | 1 | 8 Jam |

**Tabel 3.1**

**(Lanjutan)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Deskripsi** | **Aktor** | **Prioritas** | **Sprint** | **Waktu** |
| PB - 4 | Fitur pencarian barang berdasarkan nama produk agar proses pencarian barang menjadi lebih efisien. | Customer | 2 | 1 | 6 Jam |
| PB - 5 | Menambah dan mengatur produk di cart agar dapat mengumpulkan barang yang ingin dibeli sebelum melakukan pembayaran. | Customer | 1 | 1 | 10 Jam |
| PB - 6 | Melakukan checkout dan pembayaran agar dapat menyelesaikan transaksi pembelian. | Customer | 1 | 1 | 12 Jam |
| PB - 7 | Melihat status pesanan dan riwayat transaksi agar bisa memantau progres produk pembelian. | Customer | 2 | 1 | 10 Jam |
| PB - 8 | Melihat informasi toko seperti About Us, Kontak, Syarat dan Ketentuan, dan lain - lain agar customer bisa tahu informasi mengenai toko. | Customer | 3 | 1 | 8 Jam |
| PB - 9 | Melihat informasi bantuan seperti FAQ, Pengiriman, dan lain - lain agar customer mengetahui sistem kerja toko. | Customer | 3 | 1 | 8 Jam |
| PB - 10 | Fitur notifikasi agar customer selalu mendapat informasi terbaru mengenai produk jualan atau pesanan customer. | Customer | 1 | 1 | 8 Jam |
| PB - 11 | Mengatur profile agar dapat mengelola informasi pribadi seperti alamat, password, dan lain - lain. | Customer | 1 | 1 | 12 Jam |

**Tabel 3.1**

**(Lanjutan)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Deskripsi** | **Aktor** | **Prioritas** | **Sprint** | **Waktu** |
| PB - 12 | Mengelola stok produk secara real-time agar stok barang yang ditampilkan ke customer selalu akurat. | Staff | 1 | 2 | 14 Jam |
| PB - 13 | Memproses permintaan pengembalian produk agar proses retur barang dapat tercatat dan diselesaikan sesuai prosedur. | Staff | 3 | 2 | 12 Jam |
| PB - 14 | Menginput nomor resi pengiriman agar customer dapat melacak status pengiriman mereka. | Staff | 2 | 2 | 6 Jam |
| PB - 15 | Menambah dan mengelola data produk agar katalog produk yang ditampilkan kepada customer selalu akurat. | Admin | 1 | 2 | 12 Jam |
| PB - 16 | Melihat laporan transaksi, informasi inventori, dan data pengguna agar dapat membuat keputusan bisnis yang terstruktur. | Admin | 2 | 2 | 12 Jam |
| PB - 17 | Menambah dan mengelola data voucher. | Admin | 2 | 2 | 8 Jam |
| PB - 18 | Menambah, mengelola, dan mengaktifasikan akun staff dan admin. | Admin | 1 | 2 | 12 Jam |
| PB - 19 | Menambah dan mengelola kategori produk dan unit konversi. | Admin | 1 | 2 | 14 Jam |

**3.3 Sprint Backlog**

Setelah Product Backlog Item dipilih, tim pengembang akan memecahnya menjadi tugas-tugas teknis yang lebih kecil agar dapat dikerjakan dengan teratur. Proses ini menghasilkan Sprint Backlog, yang pada dasarnya adalah rencana kerja harian bagi tim pengembang selama Sprint berlangsung. Tim pengembang dapat menambah, mengubah, atau menghapus tugas sesuai kebutuhan untuk mencapai Sprint Goal. Tabel dibawah ini akan memberikan penjelasan lebih jelas mengenai Sprint Backlog dari sistem ini :

**Tabel 3.2**

**Tabel Sprint Backlog**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Product Backlog** | **Deskripsi** | **Tugas** |
| SP - 1 | PB - 1 | (SP-1.1) Desain skema database untuk User, Produk, StockConfig, Voucher, Cart, Order, Return, StockMovement. | Egbert |
| SP - 2 | PB - 2 | (SP-2.1) Inisialisasi proyek backend Node.js & koneksi DB. | Given |
| (SP-2.2) Buat API endpoint untuk registrasi dan login user. | Given |
| (SP-2.3) Inisialisasi proyek frontend React.js & Tailwind CSS/Shadcn. | Aaron |
| (SP-2.4) Buat halaman UI registrasi dan rogin. | Aaron |
| SP - 3 | PB - 3 | (SP-3.1) Buat API untuk mengambil semua data produk. | Given |
| (SP-3.2) Buat komponen UI untuk menampilkan produk di katalog. | Aaron |
| SP - 4 | PB - 4 | (SP-4.1) Buat logika fitur pencarian di backend. | Christopani |
| (SP-4.2) Buat halaman UI untuk detail produk. | Aaron |
| SP - 5 | PB - 5 | (SP-5.1) Buat API endpoint untuk mengelola cart (tambah, hapus, update kuantitas). | Christopani |
| (SP-5.2) Implementasi UI halaman cart dan hubungkan ke API. | Aaron |
| SP - 6 | PB - 6 | (SP-6.1) Buat API endpoint untuk proses checkout. | Given |
| (SP-6.2) Sambungkan API RajaOngkir di backend untuk menghitung ongkos kirim. | Given |
| (SP-6.3) Buat halaman UI checkout. | Aaron |

**Tabel 3.2**

**(Lanjutan)**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Product Backlog** | **Deskripsi** | **Tugas** |
| SP - 7 | PB - 7 | (SP-7.1) Buat API endpoint untuk mengambil riwayat pesanan dan status pesanan customer. | Christopani |
| (SP-7.2) Buat Halaman UI Pesanan Saya (Riwayat & Status) di dasboard Customer. | Aaron |
| SP - 8 | PB - 8 | (SP-8.1) Buat Halaman UI untuk About Us. | Aaron |
| (SP-8.2) Buat Halaman UI untuk Kontak Toko. | Christopani |
| (SP-8.3) Buat Halaman UI untuk Syarat dan Ketentuan Toko. | Aaron |
| (SP-8.4) Buat Halaman UI untuk Kebijakan Privasi. | Christopani |
| SP - 9 | PB - 9 | (SP-9.1) Buat Halaman UI untuk FAQ. | Aaron |
| (SP-9.2) Buat Halaman UI untuk Pengiriman yang digunakan Toko. | Christopani |
| (SP-9.3) Buat Halaman UI untuk Pengembalian barang toko. | Aaron |
| (SP-9.4) Buat Halaman UI untuk Pembayaran yang digunakan toko. | Christopani |
| SP - 10 | PB - 10 | (SP-10.1) Buat API Endpoint untuk mengambil notifikasi milik user tertentu. | Christopani |
| (SP-10.2) Implementasi trigger notifikasi otomatis. | Given |
| (SP-10.3) Buat UI list notifikasi di dashboard Customer. | Aaron |
| SP - 11 | PB - 11 | (SP-11.1) Buat API Endpoint untuk mengambil profile dari user yang sedang login. | Christopani |
| (SP-11.2) Buat API Endpoint untuk mengedit profile dari user yang sedang login. | Given |
| (SP-11.3) Buat API Endpoint untuk mengubah password dari user yang sedang login. | Aaron |
| (SP-11.4) Buat halaman UI untuk profile saya. | Aaron |

**Tabel 3.2**

**(Lanjutan)**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Product Backlog** | **Deskripsi** | **Tugas** |
| SP - 11 | PB - 11 | (SP-11.5) Buat form untuk mengedit profile user yang sedang login. | Christopani |

**3.4 Pembuktian Sprint**

Pada sub-bab ini, akan diberikan bukti implementasi dari setiap Product Backlog yang telah dibuat. Pembuktian disajikan dalam tiga bentuk, yaitu: alur logika (algoritma), segmen kode program, database yang terhubung dan bentuk gambar dari UI sistem yang telah berhasil dibuat.

1. **SP - 1 (Database)**

Pada Algoritma 3.1, sistem memulai proses inisialisasi data produk dengan menetapkan identitas dasar berupa nama produk (02), slug unik untuk URL (03), serta kategori produk tersebut (04). Selanjutnya, sistem mengisi detail komersial dengan menetapkan satuan unit (05), harga jual (06), dan merek produk (07). Informasi terkait promosi juga diatur dengan menyimpan persentase diskon (08) beserta batas waktu berlakunya (09). Kemudian, sistem mengelola data inventaris dengan memasukkan jumlah stok saat ini (10), batas stok minimum untuk peringatan (11), serta ketersediaan unit (12). Sistem juga melengkapi informasi visual dan deskriptif melalui daftar gambar (13) dan deskripsi lengkap produk (14). Data kinerja produk turut dicatat, meliputi rata-rata rating (15), jumlah ulasan (16), total produk terjual (17), serta jumlah dilihat (18). Untuk spesifikasi teknis, sistem menyimpan attribut rinci seperti diameter dalam inci (19) dan milimeter (20), tipe produk (21), panjang dalam meter (22), serta ketersediaan fitur mof (23). Akhirnya, sistem menetapkan stempel waktu pembuatan (24) dan pembaruan (25) ke tanggal saat ini, lalu mengatur status operasional produk, yakni apakah aktif (26) dan apakah produk tersebut merupakan produk unggulan (featured) (27).

**Algoritma 3.1 Contoh Algoritma Database**

1. Set product.\_id = generated ObjectId

**Algoritma 3.1 (Lanjutan)**

1. Set product.name = name
2. Set product.slug = slug
3. Set product.category = category
4. Set product.unit = unit
5. Set product.price = price
6. Set product.brand = brand
7. Set product.discount.percentage = discountPercentage
8. Set product.discount.validUntil = discountValidUntil
9. Set product.stock = stock
10. Set product.minStock = minStock
11. Set product.availableUnits = availableUnits
12. Set product.images = images
13. Set product.description = description
14. Set product.rating.average = ratingAverage
15. Set product.rating.count = ratingCount
16. Set product.sold = sold
17. Set product.views = views
18. Set product.attributes.diameter\_inch = diameterInch
19. Set product.attributes.diameter\_mm = diameterMm
20. Set product.attributes.type = type
21. Set product.attributes.length\_meter = lengthMeter
22. Set product.attributes.has\_mof = hasMof
23. Set product.createdAt = current date
24. Set product.updatedAt = current date
25. Set product.isActive = isActive
26. Set product.isFeatured = isFeatured

Pada Segmen Program 3.1, sistem akan menyimpan informasi lengkap mengenai produk, sepeti "Fumato Pipa PVC AW 1 inc", dimulai dari identitas unik (\_id) hingga informasi komersial seperti harga (price), kategori, dan merek (02-10). Data ini juga mencakup informasi stok saat ini (stock) serta batas minimum stok (minStock) (15 – 16). Data ini juga menangani aspek visual dengan menyimpan URL gambar yang terintegrasi dengan layanan Cloudinary. Terakhir, terdapat isActive dan createdAt yang digunakan sistem untuk mengontrol visibilitas produk di katalog serta keperluan audit data.

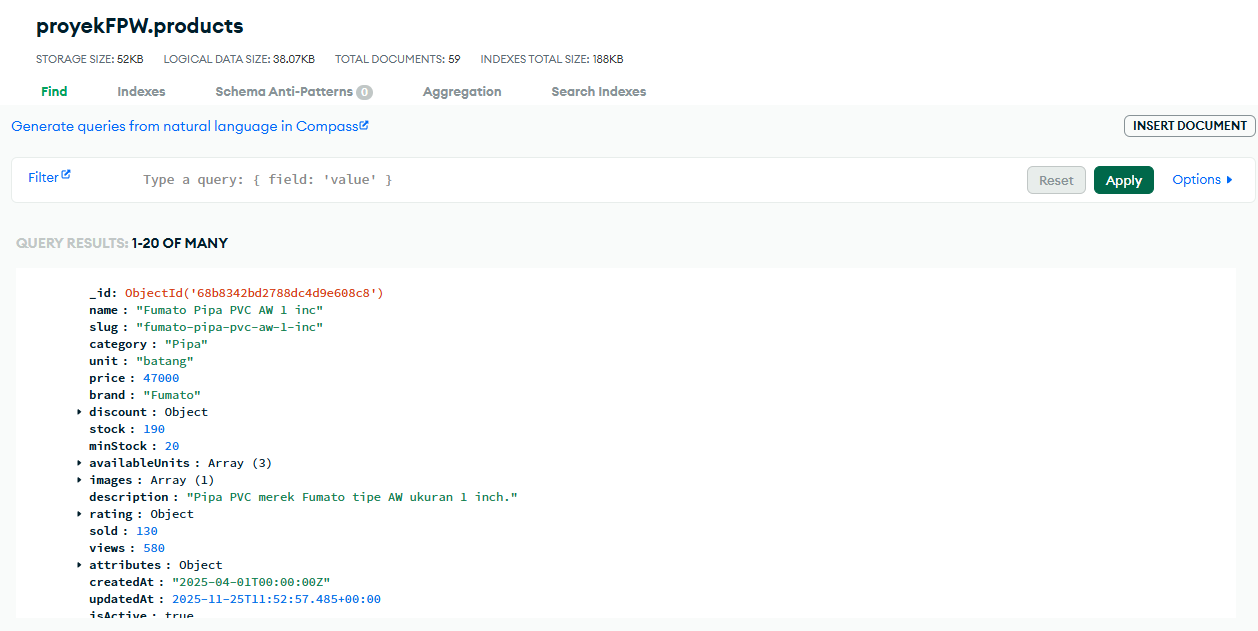
**Segmen Program 3.1 Contoh Program Database**

1. {
2. "\_id": {
3. "$oid": "68b8342bd2788dc4d9e608c8"
4. },
5. "name": "Fumato Pipa PVC AW 1 inc",
6. "slug": "fumato-pipa-pvc-aw-1-inc",

**Segmen Program 3.1 (Lanjutan)**

1. "category": "Pipa",
2. "unit": "batang",
3. "price": 47000,
4. "brand": "Fumato",
5. "discount": {
6. "percentage": 0,
7. "validUntil": ""
8. },
9. "stock": 200,
10. "minStock": 20,
11. "availableUnits": [
12. "batang",
13. "meter",
14. "pcs"
15. ],
16. "images": [
17. "https://res.cloudinary.com/dls3io1p0/image/upload/v1761719958/F
18. umato\_Pipa\_PVC\_AW\_1\_inc\_zqocec.png"
19. ],
20. "description": "Pipa PVC merek Fumato tipe AW ukuran 1 inch.",
21. "rating": {
22. "average": 4.3,
23. "count": 45
24. },
25. "sold": 120,
26. "views": 580,
27. "attributes": {
28. "diameter\_inch": "1",
29. "diameter\_mm": 32,
30. "type": "AW",
31. "length\_meter": 4,
32. "has\_mof": true
33. },
34. "createdAt": "2025-04-01T00:00:00Z",
35. "updatedAt": "2025-04-01T00:00:00Z",
36. "isActive": true,
37. "isFeatured": false
38. },
39. …

**Bentuk Gambar :**

****

**Gambar 3.2**

**Bentuk Gambar Database**

1. **SP - 2 (Login dan Register)**

Pada Algoritma 3.2, sistem mengambil email (01) dan password (02) yang dimasukkan oleh pengguna. Selanjutnya, sistem memeriksa apakah email tersebut terdaftar di database (03). Jika email ditemukan, maka sistem akan memeriksa apakah akun tersebut aktif (04). Jika akun aktif, sistem kemudian akan membandingkan password yang dimasukkan dengan password yang tersimpan di database (05). Jika password cocok (06), maka sistem akan menampilkan pesan “Login Successful” (07), memperbarui tanggal login terakhir dengan tanggal saat ini (08), dan mengalihkan pengguna ke halaman Dashboard (09). Namun, jika password tidak cocok, sistem akan menampilkan pesan “Email atau password salah” (10–11). Apabila akun tidak aktif, sistem akan menampilkan pesan “Akun tidak aktif. Hubungi administrator.” (13–14). Jika sejak awal email tidak ditemukan di database (16), maka sistem juga akan menampilkan pesan “Email atau password salah” (17). Terakhir, pada langkah 19, algoritma berakhir setelah seluruh proses pengecekan selesai dilakukan, baik login berhasil maupun gagal.

**Algoritma 3.2 Contoh Algoritma Login**

1. Get email
2. Get password
3. IF (email exists in database) THEN
4. IF (account is active) THEN
5. Compare entered password with stored password
6. IF (password matches) THEN
7. Display “Login Successful”
8. Update lastLogin = current date
9. Redirect to Dashboard Screen
10. ELSE
11. Display “Email atau password salah”
12. END IF
13. ELSE
14. Display “Akun tidak aktif. Hubungi administrator.”
15. END IF
16. ELSE
17. Display “Email atau password salah”
18. END IF
19. End

Pada Algoritma 3.3, sistem akan mengambil input dari pengguna (01) berupa username, email, password, nama lengkap, nomor telepon, dan alamat. Setelah semua data dimasukkan, sistem akan memeriksa apakah username yang diberikan sudah digunakan oleh pengguna lain (02). Jika ternyata sudah ada, maka sistem akan menampilkan pesan “Username sudah digunakan” (03) agar pengguna memilih username lain. Jika username masih tersedia, sistem kemudian memeriksa apakah email yang dimasukkan sudah terdaftar (04). Jika email sudah digunakan oleh akun lain, sistem akan menampilkan pesan “Email sudah terdaftar” (05). Namun, jika baik username maupun email belum digunakan, sistem akan melanjutkan proses dengan mengenkripsi password (07) agar disimpan dalam bentuk aman di database. Setelah itu, sistem akan membuat data pengguna baru (08–10) yang berisi username, email, password yang telah di-hash, nama lengkap, nomor telepon, alamat, serta menetapkan role sebagai “user” dan status akun aktif (isActive = true). Setelah semua data berhasil disimpan, sistem akan menampilkan pesan “Registrasi berhasil” (11) sebagai tanda bahwa proses pendaftaran akun telah selesai, dan algoritma berakhir pada langkah 13 (End).

**Algoritma 3.3 Contoh Algoritma Register**

1. Get username, email, password, full name, phone, address
2. IF (username already exists) THEN
3. Display “Username sudah digunakan”
4. ELSE IF (email already exists) THEN
5. Display “Email sudah terdaftar”
6. ELSE
7. Encrypt (password) → hashedPassword
8. Create new user record with:
9. username, email, hashedPassword, full name, phone, address,
10. role = “user”, isActive = true
11. Display “Registrasi berhasil”
12. END IF
13. End

Pada Segmen Program 3.2, sistem akan mengambil user dari database berdasarkan inputan email (01). Akan ada pengecekan seperti, jika user tersebut tidak ditemukan di database, maka akan dilemparkan pesan error (02 - 04) dan juga apakah user yang login masih aktif atau tidak, jika tidak aktif, maka akan dilemparkan pesan error (05 - 08). Kemudian akan dilakukan pengecekan password apakah sesuai dengan hashed password yang ada di database, jika tidak cocok maka akan di lemparkan error juga (09 - 13). Jika semua progress tersebut selesai tanpa masalah maka last login user akan di isi dengan waktu sekarang (14) dan akan di simpan perubahannya (15). Terakhir akan di kirimkan object dari attribut user seperti id, email, nama, username, dan lain – lain ke frontend (16 - 24).

**Segmen Program 3.2 Contoh Program Login**

1. const user = await User.findOne({ email: credentials.email });
2. if (!user) {
3. throw new Error('Email atau password salah');
4. }
5. if (!user.isActive) {
6. throw new Error('Akun Anda tidak aktif. Hubungi
7. administrator.');
8. }
9. const isValidPassword = await compare(credentials.password,
10. user.password);
11. if (!isValidPassword) {
12. throw new Error('Email atau password salah');
13. }
14. user.lastLogin = new Date();
15. await user.save();

**Segmen Program 3.2 (Lanjutan)**

1. return {
2. id: String(user.\_id),
3. email: user.email,
4. name: user.fullName,
5. username: user.username,
6. role: user.role,
7. phone: user.phone,
8. isActive: user.isActive,
9. };

Pada Segmen Program 3.3, sistem akan mencari user dengan username yang diinputkan oleh user, jika ternyata ada user yang usernamenya mirip maka akan dilemparkan pesan error (02 - 09). Hal yang sama juga di lakukan untuk pengecekan email, jika ada user yang emailnya mirip maka akan dilemparkan pesan error juga (10 - 16). Ketika seluruh proses tersebut selesai maka inputan password akan di hash untuk keamanan (17) dan user baru akan di buat berdasarkan inputan user (18 - 27) dan kemudian akan dikirimkan hasilnya beserta pesan ke frontend (28 - 37).

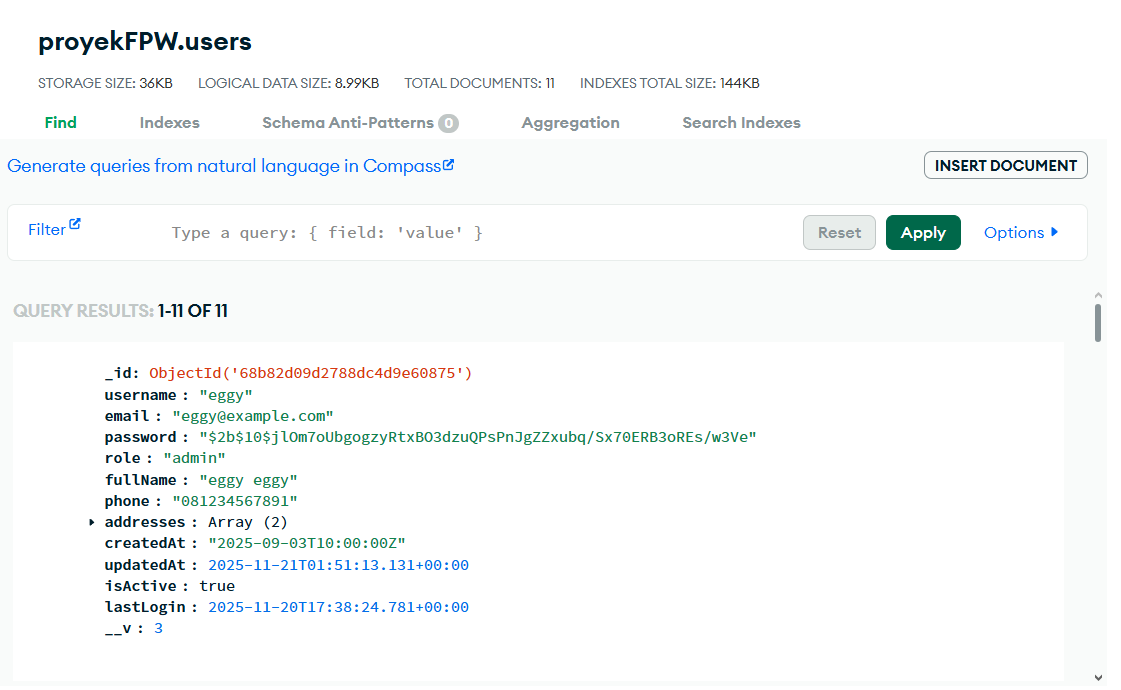
**Segmen Program 3.3 Contoh Program Register**

1. …
2. const existingUsername = await User.findOne({ username:
3. input.username});
4. if (existingUsername) {
5. throw new TRPCError({
6. code: 'CONFLICT',
7. message: 'Username sudah digunakan',
8. });
9. }
10. const existingEmail = await User.findOne({ email: input.email });
11. if (existingEmail) {
12. throw new TRPCError({
13. Code: 'CONFLICT',
14. message: 'Email sudah terdaftar',
15. });
16. }
17. const hashedPassword = await hash(input.password, 10);
18. const user = await User.create({
19. username: input.username,
20. email: input.email,
21. password: hashedPassword,
22. fullName: input.fullName,
23. phone: input.phone,
24. address: input.address,
25. role: 'user', // Default role for new registration

**Segmen Program 3.3 (Lanjutan)**

1. isActive: true,
2. });
3. return {
4. success: true,
5. message: 'Registrasi berhasil! Silakan login.',
6. user: {
7. id: String(user.\_id),
8. username: user.username,
9. email: user.email,
10. fullName: user.fullName,
11. },
12. };
13. …

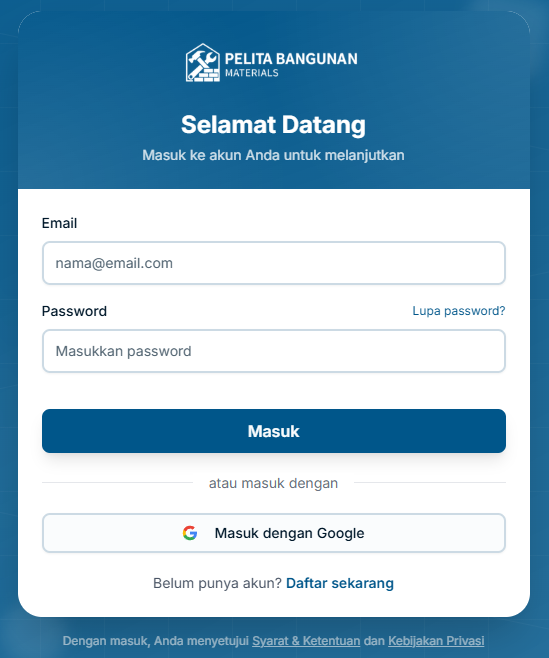
**Database yang Digunakan :**



**Gambar 3.3**

**Gambar Database Login dan Register**

**Bentuk Gambar :**



**Gambar 3.4**

**Bentuk Gambar Login Register**

1. **SP - 3 (Katalog Produk dan Detail Produk)**

Pada Algoritma 3.4, sistem akan mengambil parameter input (01) seperti query untuk menentukan filter pencarian produk, skip untuk melewati sejumlah data di awal (berguna untuk pagination), limit untuk membatasi jumlah produk yang ditampilkan per halaman, serta sortQuery untuk mengatur urutan hasil (misalnya berdasarkan harga atau tanggal). Selanjutnya, sistem akan mengambil data produk dari database (02) dengan perintah Product.find(query) dan menambahkan metode .skip(input.skip), .limit(input.limit), .sort(sortQuery), serta .lean() untuk mengoptimalkan hasil agar lebih efisien. Hasil pengambilan data tersebut kemudian disimpan dalam variabel products (03 - 08). Setelah itu, sistem juga akan menghitung jumlah total produk yang sesuai dengan query pencarian (09) dan menyimpannya dalam variabel total. Kemudian, sistem akan mengembalikan dua data utama (10), yaitu products yang berisi daftar produk yang ditemukan dan total yang berisi jumlah keseluruhan produk. Langkah berikutnya adalah menampilkan daftar produk tersebut pada halaman Katalog (11) dengan menerapkan fitur pagination agar pengguna dapat melihat produk halaman demi halaman. Terakhir, proses algoritma berakhir setelah semua data berhasil ditampilkan (12).

**Algoritma 3.4 Contoh Algoritma Katalog Produk**

1. Get query, skip, limit, and sortQuery
2. Retrieve products from database using:
3. Product.find(query)
4. .skip(input.skip)
5. .limit(input.limit)
6. .sort(sortQuery)
7. .lean()
8. Store the result in products
9. Count total number of products → total
10. Return: products, total
11. Display product list on Catalog Page with pagination
12. End

Pada Algoritma 3.5, sistem akan terhubung ke database untuk mengakses data produk yang tersimpan. Setelah itu, sistem akan mencari produk di database (01 - 03) dengan kondisi bahwa slug harus sama dengan input.slug dan isActive bernilai true, artinya hanya produk yang masih aktif dan tersedia yang akan ditampilkan. Jika produk ditemukan (04), maka sistem akan menampilkan detail produk (05 - 06) seperti nama, harga, deskripsi, gambar, dan informasi lainnya di halaman detail produk. Namun, jika produk tidak ditemukan, sistem akan menampilkan pesan “Product not found” (07 - 09) sebagai tanda bahwa slug yang diminta tidak cocok dengan produk manapun di database. Terakhir, algoritma berakhir setelah proses pengecekan dan penampilan data selesai dilakukan (10).

**Algoritma 3.5 Contoh Algoritma Detail Produk**

1. Find product where:
2. slug = input.slug
3. and isActive = true
4. IF (product exists) THEN
5. Display product details (name, price, description, image,
6. etc.0029
7. ELSE
8. Display “Product not found”
9. END IF
10. End

Pada Segmen Program 3.4, sistem akan mengambil produk dari database berdasarkan query dan juga inputan - inputan tambahan seperti skip, limit, sort, dan juga lean (02 - 06). Kemudian hasil filter produknya akan dikembalikan beserta jumlah filternya ke frontend (07 - 10).

**Segmen Program 3.4 Contoh Program Katalog Produk**

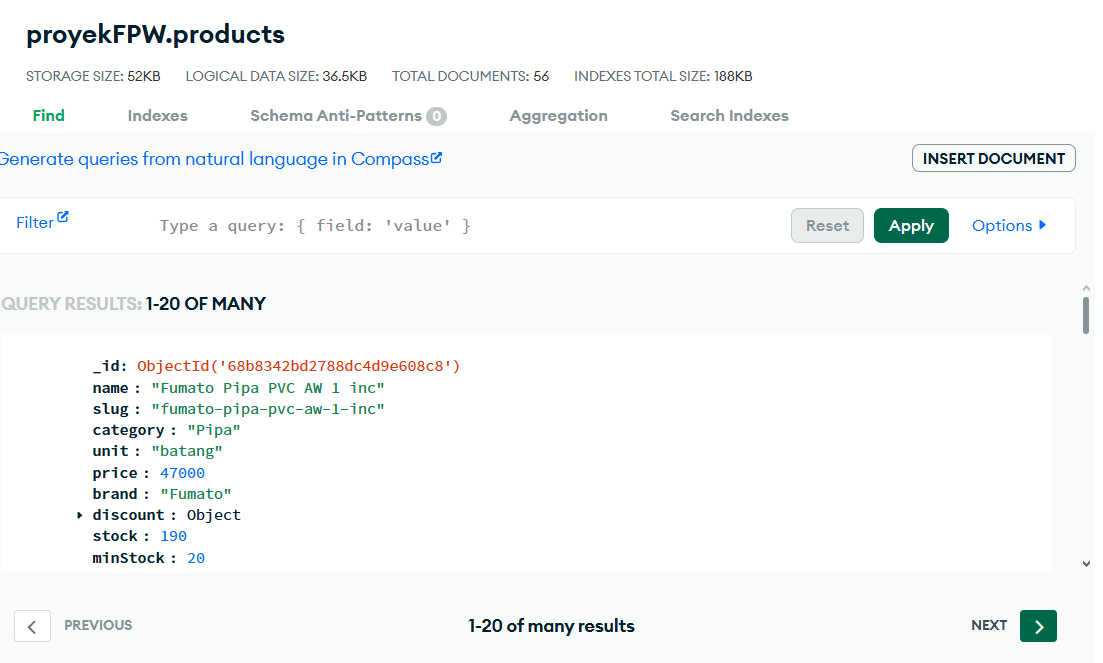
1. …
2. const products = await Product.find(query)
3. .skip(input.skip)
4. .limit(input.limit)
5. .sort(sortQuery)
6. .lean<IProductData[]>();
7. return {
8. products,
9. total,
10. };
11. …

Pada Segmen Program 3.5, sistem akan mengambil detail produk dari database berdasarkan slug yang diinputkan dan juga produknya harus masih aktif (02 - 05). Jika produknya tidak ditemukan di database maka akan dikirimkan pesan error “Product not found” (06 - 08). Kemudian jika produknya sudah ditemukan maka akan dikembalikan ke frontend (09).

**Segmen Program 3.5 Contoh Program Detail Produk**

1. …
2. const product = await Product.findOne({
3. slug: input.slug,
4. isActive: true,
5. }).lean<IProductData>();
6. if (!product) {
7. throw new Error("Product not found");
8. }
9. return product;
10. …

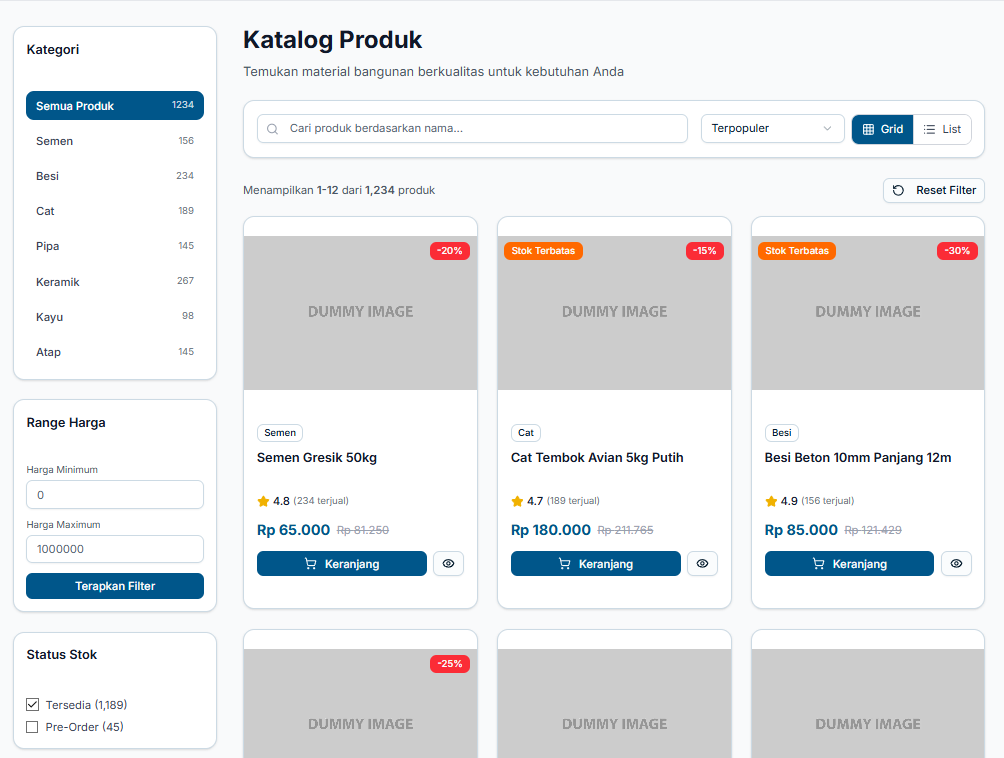
**Database yang Digunakan :**

****

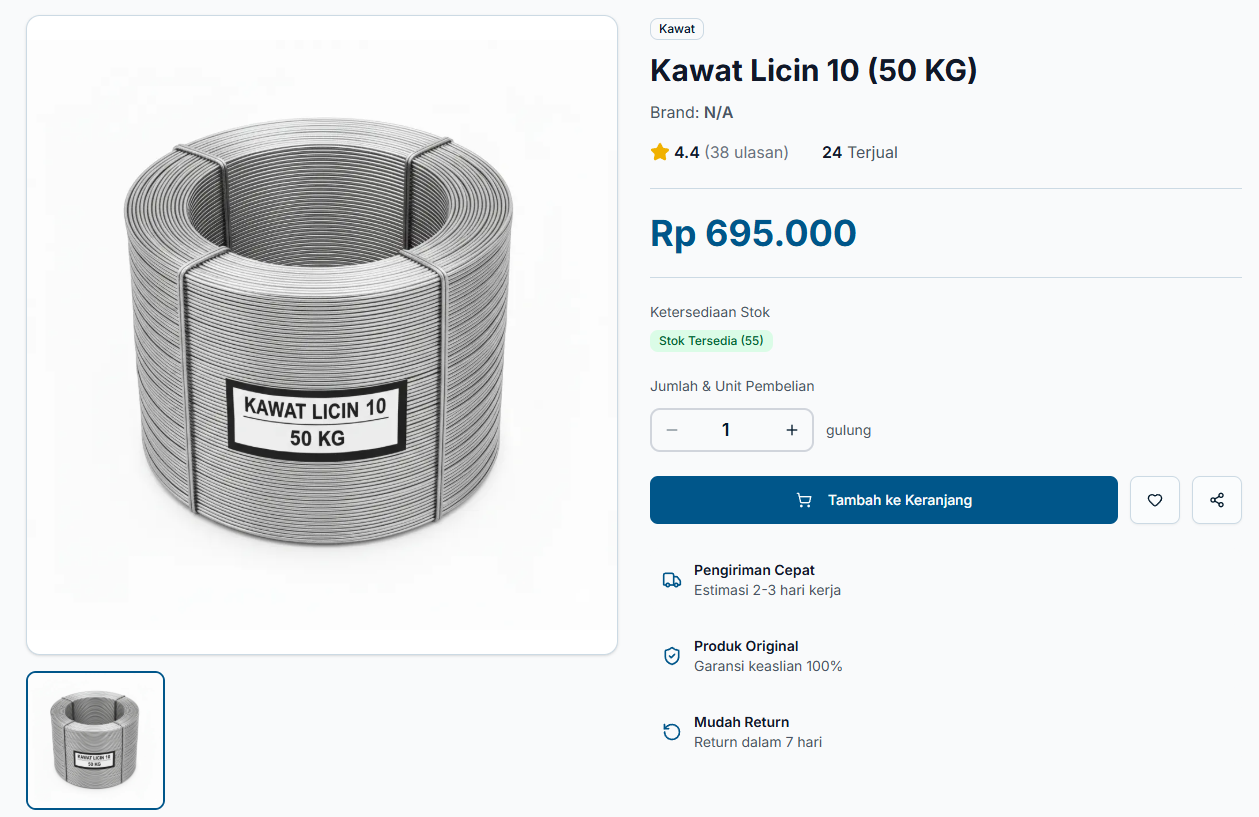
**Gambar 3.5**

**Gambar Database Katalog dan Detail Produk**

**Bentuk Gambar :**



**(a)**



**(b)**

**Gambar 3.6**

**Bentuk Gambar (a) Katalog Produk. (b) Detail Produk**

1. **SP - 4 (Fitur Pencarian Barang)**

Pada Algoritma 3.6, sistem akan mengambil input pencarian dari pengguna (01) seperti kategori produk, harga minimum (minPrice), harga maksimum (maxPrice), status diskon (hasDiscount), dan kata kunci pencarian (search keyword). Setelah itu, sistem akan menginisialisasi objek query (02) yang nantinya digunakan untuk menampung semua kondisi pencarian. Jika pengguna memasukkan kategori (03 - 04), maka sistem akan menambahkan filter kategori ke dalam query. Kemudian, jika pengguna mengisi batas harga minimum atau maksimum (05 - 06), sistem akan membuat objek harga di dalam query. Jika harga minimum diisi (07 - 08), maka sistem akan menetapkan query.price.gte = input.minPrice, yang berarti produk dengan harga lebih besar atau sama dengan nilai tersebut akan diambil. Begitu pula jika harga maksimum diisi (09 - 10), maka query.price.lte = input.maxPrice, artinya produk dengan harga lebih kecil atau sama dengan nilai tersebut akan disertakan. Selanjutnya, jika pengguna mengaktifkan filter diskon (11 - 12), sistem akan menambahkan kondisi agar hanya menampilkan produk yang memiliki nilai discount.percentage lebih besar dari nol. Jika pengguna memasukkan kata kunci pencarian (13 - 16), maka sistem akan menambahkan kondisi pencarian teks (case-insensitive) agar mencari kata kunci tersebut pada kolom nama maupun deskripsi produk. Setelah seluruh kondisi query selesai disusun, sistem akan mengambil produk dari database berdasarkan query tersebut (17). Terakhir, sistem akan menampilkan daftar produk yang sesuai dengan kriteria pencarian (18) kepada pengguna, dan algoritma berakhir setelah semua hasil ditampilkan (19).

**Algoritma 3.6 Contoh Algoritma Fitur Pencarian Barang**

1. Get category, minPrice, maxPrice, hasDiscount, search keyword
2. Initialize query
3. IF (category is entered):
4. Set query.category = input.category
5. IF (minPrice or maxPrice is entered):
6. Initialize query.price = {}
7. IF (minPrice is entered):
8. Set query.price.gte = input.minPrice
9. IF (maxPrice is entered):
10. Set query.price.lte = input.maxPrice
11. IF (hasDiscount = true):
12. Set query.discount.percentage > 0
13. IF (search keyword is entered):
14. Set query.or to search where:
15. name OR description contains input.search (case-
16. Insensitive)
17. Retrieve products from database using query
18. Display list of products that match search criteria
19. End

Pada Segmen Program 3.6, sistem akan mengecek apakah ada input kategori atau tidak, jika ada maka query kategori akan berubah sesuai dengan inputan kategori (02 - 04). Jika ada inputan minPrice atau maxPrice maka akan diquery berdasarkan inputan tersebut, jika minPrice yang ada maka di filter produk yang harga minimal nya berdasarkan minPrice, begitu juga dengan maxPrice (05 - 16). Jika ada inputan diskon maka produk akan diquery berdasarkan produk yang memiliki diskon saja (17 - 19). Dan juga jika ada inputan search, maka produk akan diquery berdasarkan huruf atau kata yang ada di search bar (20 - 25).

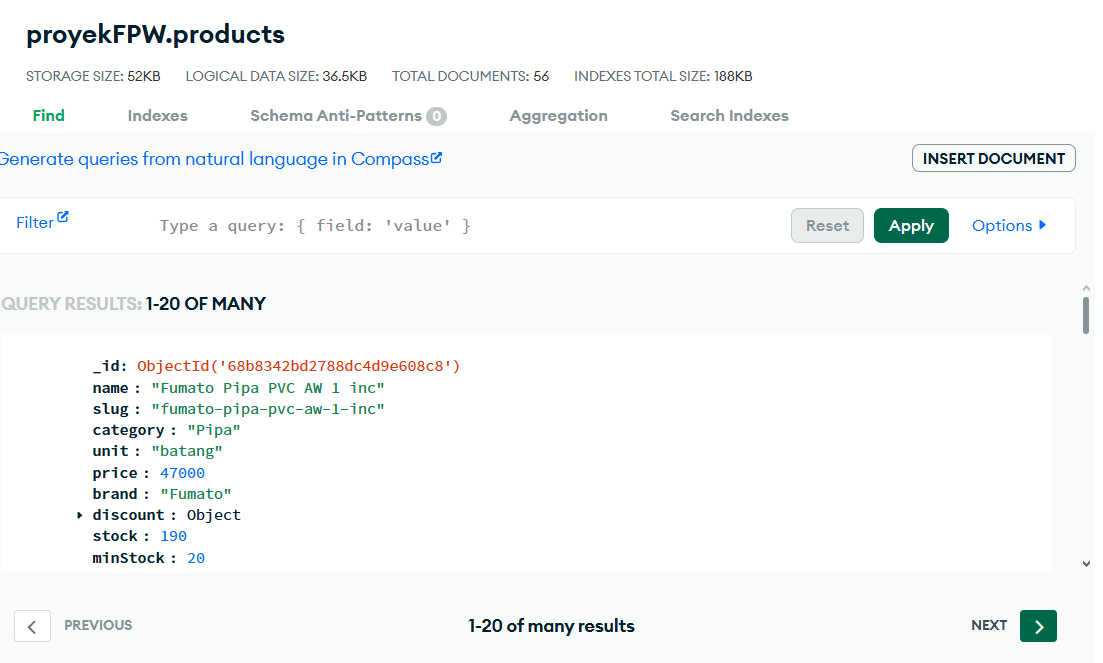
**Segmen Program 3.6 Contoh Program Fitur Pencarian Barang**

1. …

**Segmen Program 3.6 (Lanjutan)**

1. if (input.category) {
2. query.category = input.category;
3. }
4. if (input.minPrice !== undefined || input.maxPrice !==
5. undefined) {
6. query.price = {};
7. if (input.minPrice !== undefined) {
8. (query.price as Record<string, number>).$gte =
9. input.minPrice;
10. }
11. if (input.maxPrice !== undefined) {
12. (query.price as Record<string, number>).$lte =
13. Input.maxPrice;
14. }
15. }
16. if (input.hasDiscount) {
17. query["discount.percentage"] = { $gt: 0 };
18. }
19. if (input.search) {
20. query.$or = [
21. { name: { $regex: input.search, $options: "i" } },
22. { description: { $regex: input.search, $options: "i" } },
23. ];
24. }
25. …

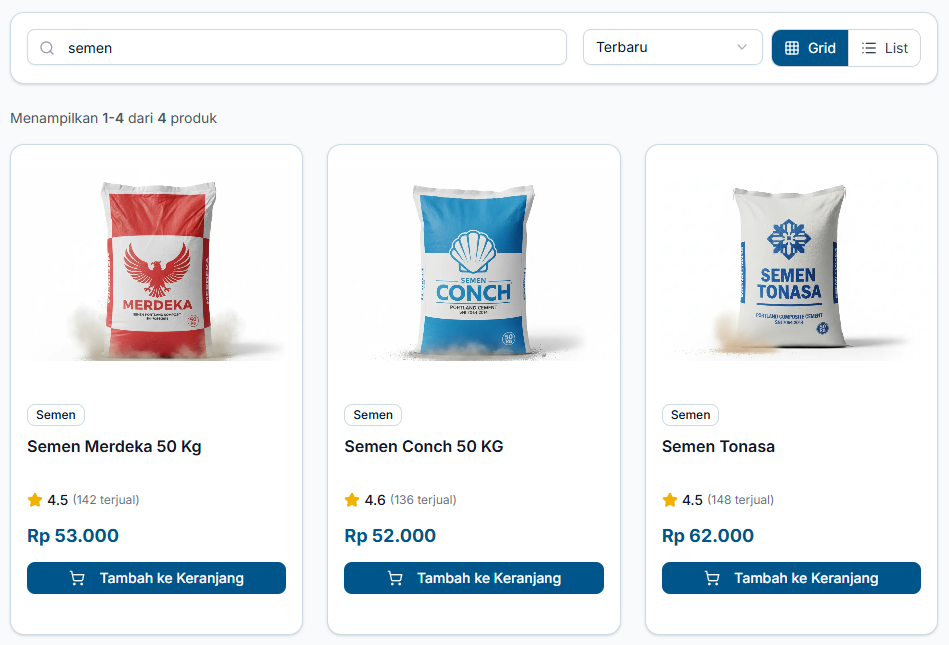
**Database yang Digunakan :**

****

**Gambar 3.7**

**Gambar Database Pencarian Barang**

**Bentuk Gambar :**



**Gambar 3.8**

**Bentuk Gambar Pencarian Barang**

1. **SP - 5 (Menambah Produk dan Mengatur Produk di Cart)**

Pada Algoritma 3.7, sistem akan mengambil userId (01) untuk mengetahui siapa pengguna yang sedang melakukan tindakan. Setelah itu, sistem akan mengambil data produk (02 - 03) yang ingin ditambahkan ke keranjang, yaitu productId, name, slug, price, quantity, unit, image, stock, dan category. Langkah berikutnya, sistem akan mencari data cart berdasarkan userId (04) untuk memastikan apakah pengguna tersebut sudah memiliki keranjang belanja. Jika cart belum ditemukan (05), maka sistem akan membuat cart baru (06) khusus untuk pengguna tersebut agar produk dapat disimpan di sana. Setelah itu, sistem akan mencari apakah produk yang sama (berdasarkan productId) sudah ada di dalam cart (08). Jika produk sudah ada (09), maka sistem tidak menambahkan item baru, tetapi menambah jumlah (quantity) produk tersebut sesuai dengan input pengguna (10) dan menampilkan pesan “Jumlah produk diperbarui di cart” (11). Namun, jika produk belum ada di cart (12), sistem akan menambahkan produk baru (13 - 15) beserta semua datanya seperti nama, harga, kuantitas, satuan, gambar, stok, dan kategori, lalu menampilkan pesan “Produk berhasil ditambahkan ke cart” (16) sebagai konfirmasi. Setelah semua perubahan selesai, sistem akan menyimpan pembaruan cart ke database (18) agar perubahan tersimpan permanen.

**Algoritma 3.7 Contoh Algoritma Menambah Produk**

1. Get userId
2. Get productId, name, slug, price, quantity, unit, image, stock,
3. category
4. Cari cart berdasarkan userId
5. IF (cart tidak ditemukan) THEN
6. Buat cart baru untuk user tersebut
7. END IF
8. Cari item di cart yang memiliki productId yang sama
9. IF (item ditemukan) THEN
10. Tambahkan quantity produk tersebut dengan input.quantity
11. Display “Jumlah produk diperbarui di cart”
12. ELSE
13. Tambahkan produk baru ke dalam cart dengan data:
14. productId, name, slug, price, quantity, unit, image,
15. Stock, category
16. Display “Produk berhasil ditambahkan ke cart”
17. END IF
18. Simpan perubahan cart (cart.save)

Pada Algoritma 3.8, sistem akan mengambil data userId (01) untuk mengetahui siapa pengguna yang sedang melakukan perubahan pada cart. Kemudian, sistem juga akan mengambil productId (02) untuk mengetahui produk mana yang ingin diperbarui, serta mengambil nilai quantity (03) yang dimasukkan oleh pengguna. Selanjutnya, sistem akan mencari cart berdasarkan userId (04) untuk memastikan pengguna tersebut memiliki keranjang belanja yang valid. Jika cart tidak ditemukan (05), maka sistem akan menampilkan pesan “Cart not found” (06) dan menghentikan proses (07) karena tidak ada cart untuk diperbarui. Namun, jika cart ditemukan, sistem akan memeriksa nilai quantity. Jika quantity bernilai 0 (09), maka sistem akan menghapus produk dari cart berdasarkan productId (10) dan menampilkan pesan “Produk dihapus dari cart” (11) sebagai tanda bahwa item tersebut telah dihapus. Sebaliknya, jika quantity lebih dari 0 (12), sistem akan mencari item di dalam cart dengan productId yang sama (13). Jika item ditemukan (14), maka sistem akan memperbarui jumlah (quantity) produk tersebut sesuai input pengguna (15) dan menampilkan pesan “Jumlah produk diperbarui” (16) sebagai konfirmasi bahwa perubahan berhasil dilakukan. Setelah semua proses selesai, sistem akan menyimpan hasil perubahan cart ke database (19) agar pembaruan tersimpan permanen, dan akhirnya algoritma berakhir setelah seluruh operasi selesai dilakukan (20).

**Algoritma 3.8 Contoh Algoritma Mengatur Produk**

1. Get userId
2. Get productId
3. Get quantity
4. Cari cart berdasarkan userId
5. IF (cart tidak ditemukan) THEN
6. Display “Cart not found”
7. End
8. END IF
9. IF (quantity == 0) THEN
10. Hapus produk dari cart berdasarkan productId
11. Display “Produk dihapus dari cart”
12. ELSE
13. Cari item di cart yang memiliki productId yang sama
14. IF (item ditemukan) THEN
15. Update quantity produk dengan input.quantity
16. Display “Jumlah produk diperbarui”
17. END IF
18. END IF
19. Simpan perubahan cart (cart.save)
20. End

Pada Segmen Program 3.7, sistem akan mencari apakah item yang ditambahkan ke cart ada atau tidak ada berdasarkan id produk dan unit produk (02 - 06). Jika ada itemnya di dalam cart maka quantity dari item tersebut ditambahkan berdasarkan quantity item yang ditambahkan ke cart (07 - 09). Tetapi jika belum ada maka item tersebut akan dipush kedalam cart, dimana attribut yang di masukkan adalah productid, name, slug, price, quantity, dan lain - lain (10 - 22).

**Segmen Program 3.7 Contoh Program Menambah Produk**

1. …
2. const existingItemIndex = cart.items.findIndex(
3. (item: ICartItem) =>
4. item.productId.toString() === input.productId &&
5. item.unit === input.unit
6. );
7. if (existingItemIndex > -1) {

**Segmen Program 3.7 (Lanjutan)**

1. cart.items[existingItemIndex].quantity +=
2. input.quantity;
3. } else {
4. cart.items.push({
5. productId: new mongoose.Types.ObjectId(input.productId),
6. name: input.name,
7. slug: input.slug,
8. price: input.price,
9. quantity: input.quantity,
10. unit: input.unit,
11. image: input.image,
12. stock: input.stock,
13. category: input.category,
14. } as ICartItem);
15. }
16. …

Pada Segmen Program 3.8, sistem akan mencari cart yang dimiliki oleh user berdasarkan userId (02 - 04). Jika cartnya tidak ditemukan maka akan diberikan pesan error “Cart not found” (05 - 10). Tetapi jika cartnya ditemukan maka akan dilihat inputan quantity yang diberikan ada berapa, jika quantitynya 0 maka item akan ditambahkan ke cart berdasarkan filter productid dan unitnya (11 - 16), tetapi jika quantitynya lebih 0, maka item yang ditemukan dalam cart akan ditambahkan quantitiynya berdasarkan inputan quantity yang diberikan (17 - 26). Kemudian hal terakhir yaitu perubahan cart tersebut akan disimpan (27).

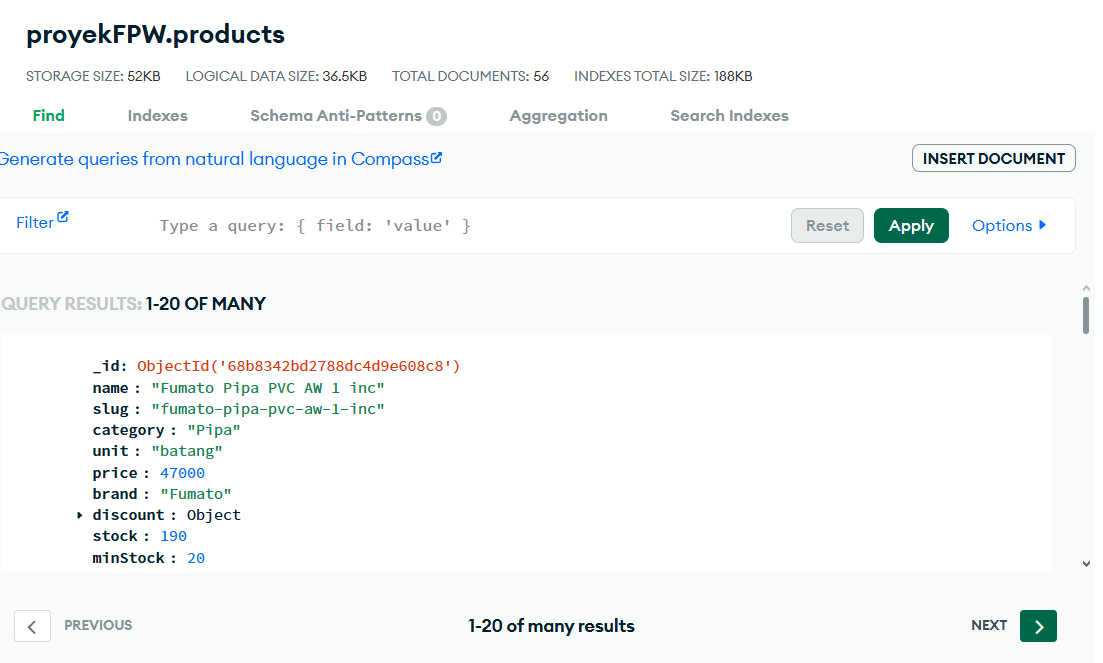
**Segmen Program 3.8 Contoh Program Mengatur Produk**

1. …
2. const cart = await Cart.findOne({
3. userId: new mongoose.Types.ObjectId(ctx.user.id)
4. });
5. if (!cart) {
6. throw new TRPCError({
7. code: 'NOT\_FOUND',
8. message: 'Cart not found',
9. });
10. }
11. if (input.quantity === 0) {
12. cart.items = cart.items.filter(
13. (item: ICartItem) =>
14. !(item.productId.toString()===input.productId &&
15. item.unit === input.unit)
16. );

**Segmen Program 3.8 (Lanjutan)**

1. } else {
2. const item = cart.items.find(
3. (item: ICartItem) =>
4. item.productId.toString() === input.productId &&
5. item.unit === input.unit
6. );
7. if (item) {
8. item.quantity = input.quantity;
9. }
10. }
11. await cart.save();
12. …

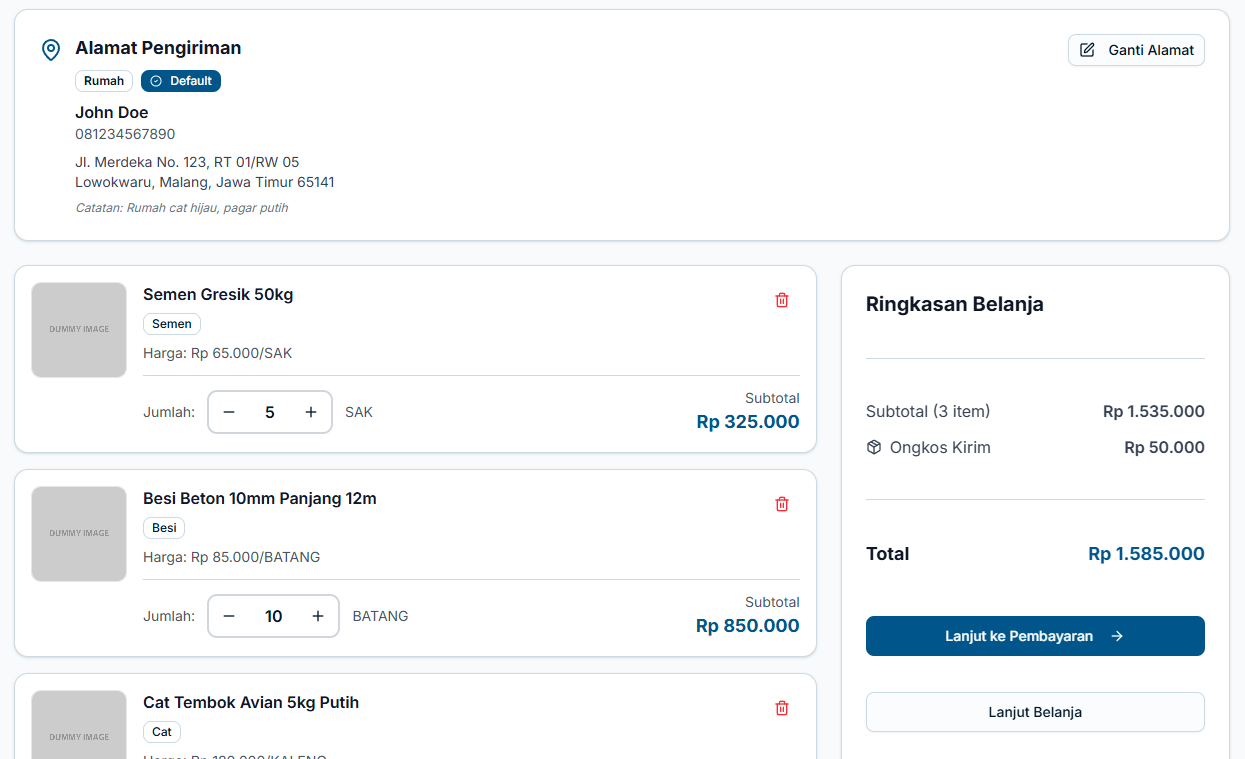
**Database yang Digunakan :**

****

**Gambar 3.9**

**Gambar Database Menambah Produk dan Mengatur Produk di Cart**

**Bentuk Gambar :**



**Gambar 3.10**

**Bentuk Gambar Menambah Produk dan Mengatur Produk di Cart**

1. **SP - 6 (Melakukan checkout dan pembayaran)**

Pada Algoritma 3.9, sistem memulai proses dengan membuat ID pesanan yang baru (01) sebagai identitas unik transaksi. Selanjutnya, sistem menyiapkan objek tanggal baru (02) dan menambahkan durasi 60 menit ke dalamnya (03) untuk menentukan batas waktu pembayaran (paymentExpiredAt). Setelah persiapan data waktu selesai, sistem akan membuat dokumen Order baru (04) yang memuat seluruh rincian transaksi. Di dalam dokumen tersebut, sistem menetapkan orderId menggunakan ID yang telah digenerate sebelumnya (05) serta menghubungkan pesanan dengan userId pengguna (06). Sistem kemudian memasukkan rincian fisik pesanan berupa daftar item (07) dan alamat pengiriman (08) yang diambil dari input. Selain itu, rincian finansial juga dicatat dengan menetapkan subtotal (09), biaya pengiriman (10), serta total tagihan (11). Metode pembayaran yang dipilih pengguna juga disimpan (12), lalu sistem mengatur status pembayaran awal menjadi 'pending' (13). Terakhir, sistem menetapkan batas waktu kedaluwarsa pembayaran sesuai perhitungan waktu sebelumnya (14) dan mengatur status pesanan menjadi 'awaiting\_payment' (15), menandakan bahwa pesanan telah siap dan sedang menunggu pembayaran dari pengguna.

**Algoritma 3.9 Contoh Algoritma Checkout**

1. GENERATE a new order ID
2. CREATE new Date object for paymentExpiredAt
3. ADD 60 minutes to paymentExpiredAt
4. CREATE new Order document with the following details:
5. Set orderId to the generated ID
6. Set userId to user's ID
7. Set items to input.items
8. Set shippingAddress to input.shippingAddress
9. Set subtotal to input.subtotal
10. Set shippingCost to input.shippingCost
11. Set total to input.total
12. Set paymentMethod to input.paymentMethod
13. Set paymentStatus to 'pending'
14. Set paymentExpiredAt to the calculated expiry time
15. Set orderStatus to 'awaiting\_payment'

Pada Algoritma 3.10, sistem memulai dengan membuat object kosong bernama updateData (01). Kemudian paymentStatus pada updateData akan diisi dengan paymentStatus dari inputan (02) dan orderStatus pada updateData juga akan diisi dengan orderStatus dari inputan (03). Jika ada inputan paidAt (04) maka paidAt pada updateData akan diisi dengan waktu saat input paidAt di berikan (05). Kemudian order akan dicari dan diupdate berdasarkan orderId (07) dimana orderId harus mirip dengan inputan (08), dan akan di update (09) dan akan dikembalikan document yang sudah diperbarui (10). Tetapi jika ordernya tidak ditemukan (11) maka akan diberikan pesan error “Order not found” (12). Terkahir akan dikembalikan pesan success dan order yang sudah diperbarui (14).

**Algoritma 3.10 Contoh Algoritma Pembayaran**

1. CREATE empty updateData object
2. SET updateData.paymentStatus to input.paymentStatus
3. SET updateData.orderStatus to input.orderStatus
4. IF input.paidAt exists THEN
5. SET updateData.paidAt to new Date from input.paidAt
6. END IF
7. FIND and UPDATE order by orderId:
8. MATCH orderId from input

**Algoritma 3.10 (Lanjutan)**

1. SET updateData
2. RETURN the new (updated) document
3. IF order not found THEN
4. THROW NOT\_FOUND error: 'Order not found'
5. END IF
6. RETURN success: true and the updated order

Pada Segmen Program 3.9, sistem akan mengenerate orderId baru (02) dan membuat paymentExpiredAt berdasarkan waktu sekarang ditambah 60 menit (03 - 06).

Kemudian order baru akan di buat dengan attribut seperti orderId yang barusan dibuat, userId, items, subtotal, paymentExpiredAt, dan attribut lainnya (07 - 19).

**Segmen Program 3.9 Contoh Program Checkout**

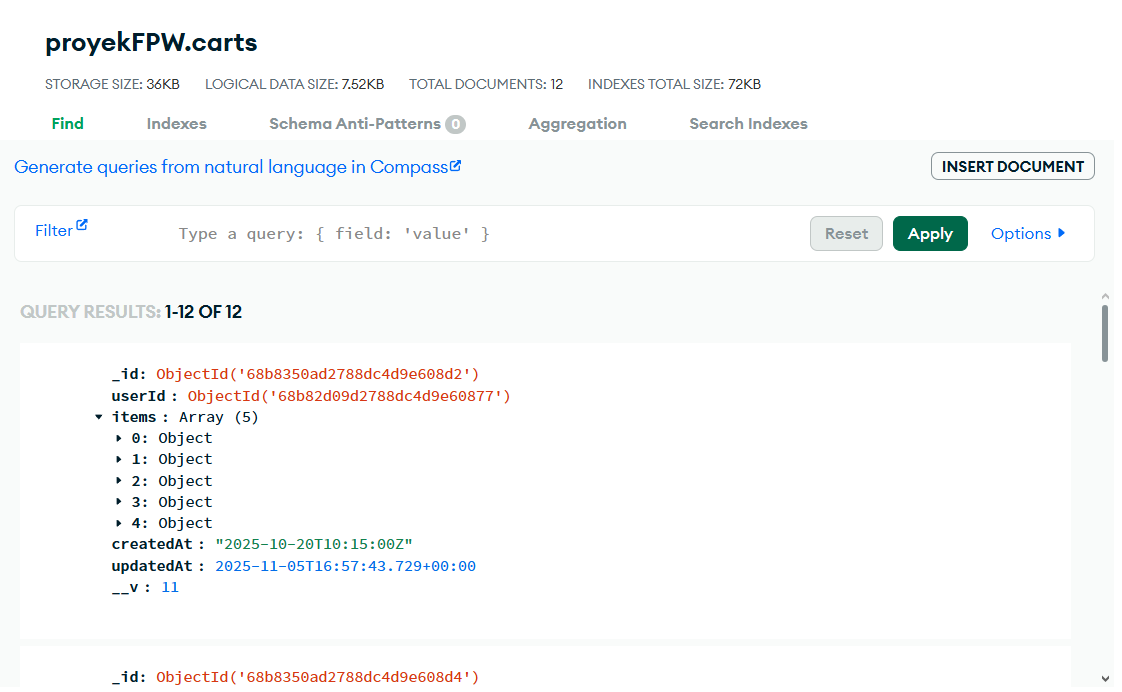
1. …
2. const orderId = generateOrderId();
3. const paymentExpiredAt = new Date();
4. paymentExpiredAt.setMinutes(
5. paymentExpiredAt.getMinutes() + 60
6. );
7. const order = await Order.create({
8. orderId,
9. userId: user.id,
10. items: input.items,
11. shippingAddress: input.shippingAddress,
12. subtotal: input.subtotal,
13. shippingCost: input.shippingCost,
14. total: input.total,
15. paymentMethod: input.paymentMethod,
16. paymentStatus: 'pending',
17. paymentExpiredAt, // 30 minutes deadline
18. orderStatus: 'awaiting\_payment',
19. });
20. …

Pada Segmen Program 3.10, sistem akan membuat updateData berdasarkan inputan paymentStatus dan orderStatus (02 - 05). Kemudian jika ada inputan paidAt maka updateData akan diisi waktu dari paidAt (06 - 09). Kemudian sistem akan mencari order dari database berdasarkan orderId dan akan mengupdate order tersebut (10 - 14). Jika order yang ingin dicari tidak ditemukan di database maka akan diberikan pesan error “Order not found” (15 - 20). Terakhir, sistem akan mengembalikan status sukses dan juga order yang sudah diupdate (21).

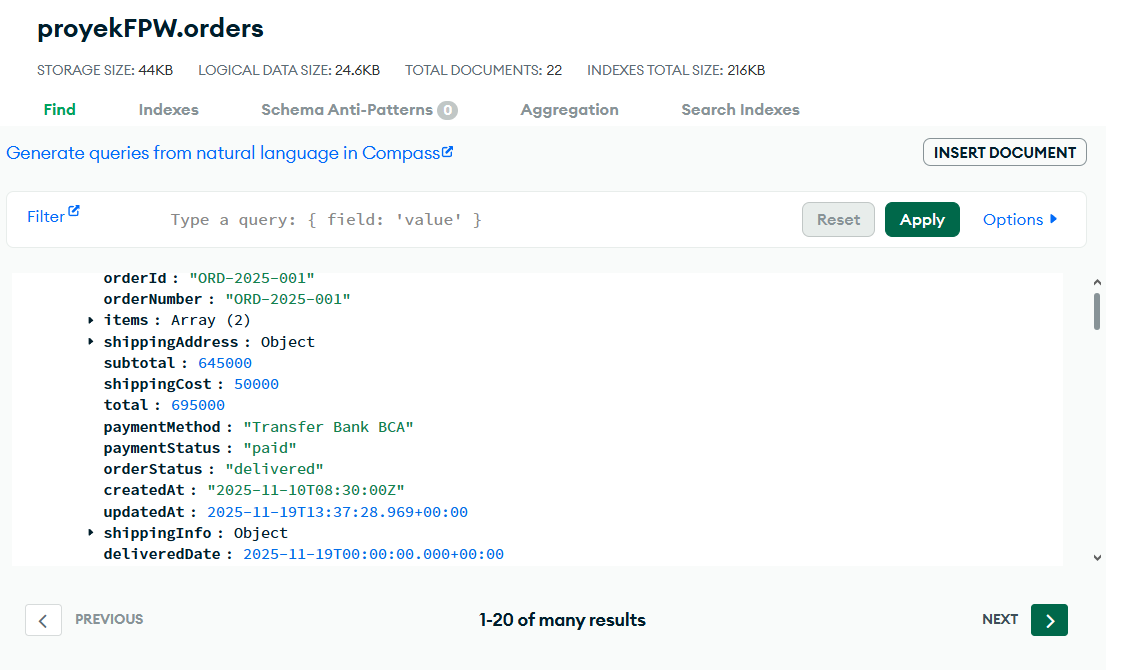
**Segmen Program 3.10 Contoh Program Pembayaran**

1. …
2. const updateData: Partial<IOrder> = {
3. paymentStatus: input.paymentStatus,
4. orderStatus: input.orderStatus,
5. };
6. if (input.paidAt) {
7. updateData.paidAt = new
8. Date(input.paidAt);
9. }
10. const order = await Order.findOneAndUpdate(
11. { orderId: input.orderId },
12. updateData,
13. { new: true }
14. );
15. if (!order) {
16. throw new TRPCError({
17. code: 'NOT\_FOUND',
18. message: 'Order not found',
19. });
20. }
21. return { success: true, order };
22. …

**Database yang Digunakan :**

****

**(a)**

****

**(b)**

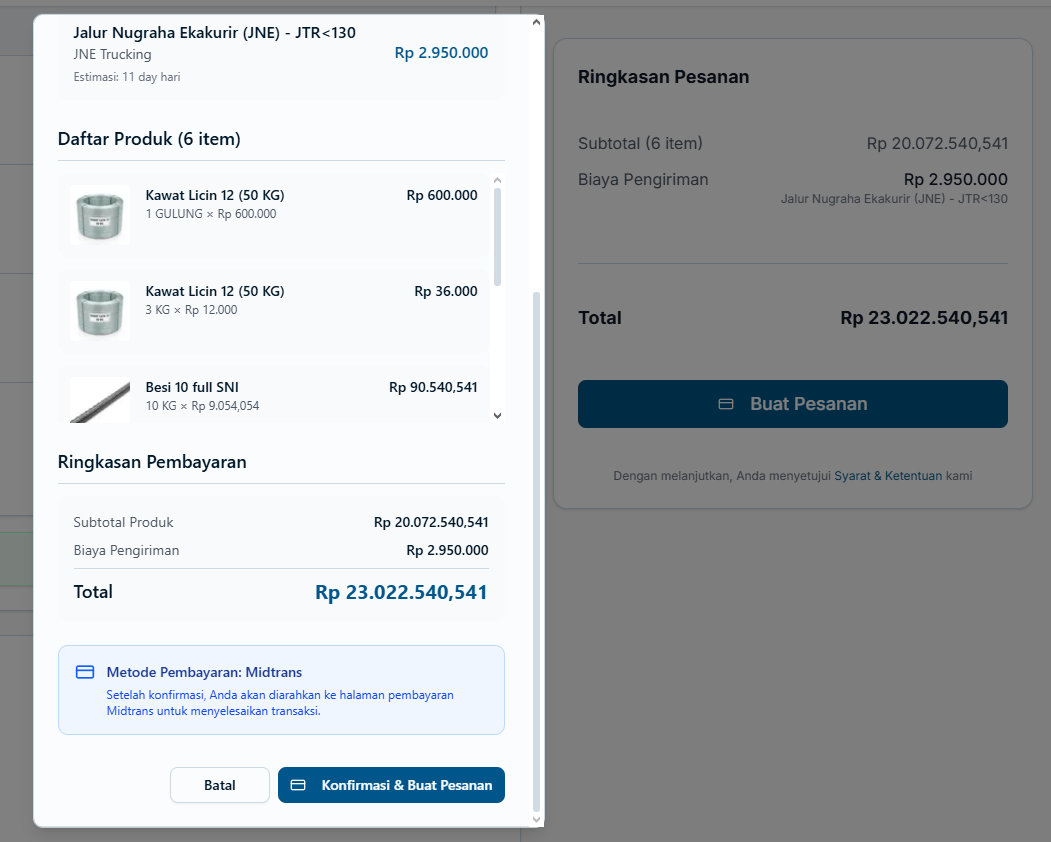
**Gambar 3.11**

**Gambar Database Checkout dan Pembayaran (a) Carts. (b) Orders**

**Bentuk Gambar :**



**(a)**



**(b)**

**Gambar 3.12**

**Bentuk Gambar (a) Checkout. (b) Pembayaran**

1. **SP - 7 (Melihat Status Pesanan dan Riwayat Transaksi)**

Pada Algoritma 3.11, sistem memulai prosedur terproteksi (01) dengan memasuki blok percobaan (02) dan melakukan koneksi ke database (03) untuk mengakses data. Selanjutnya, sistem mencari seluruh data pesanan yang memiliki userId yang sesuai dengan ID pengguna yang sedang terautentikasi (04). Agar data lebih mudah dibaca, sistem mengurutkan pesanan tersebut berdasarkan waktu pembuatan (createdAt) secara menurun (05), sehingga pesanan terbaru muncul paling atas. Setelah data ditemukan dan diurutkan, sistem mengonversi hasilnya menjadi objek JavaScript biasa (06) dan mengembalikannya kepada pengguna dalam sebuah objek (07). Namun, jika terjadi kesalahan selama proses berlangsung (08), sistem akan mencatat log kesalahan tersebut (09) untuk keperluan debugging, kemudian menghentikan proses dengan melemparkan pesan kesalahan INTERNAL\_SERVER\_ERROR dengan keterangan "Gagal mengambil data pesanan" (10) sebagai pemberitahuan kegagalan proses.

**Algoritma 3.11 Contoh Algoritma Melihat Status dan Riwayat Transaksi**

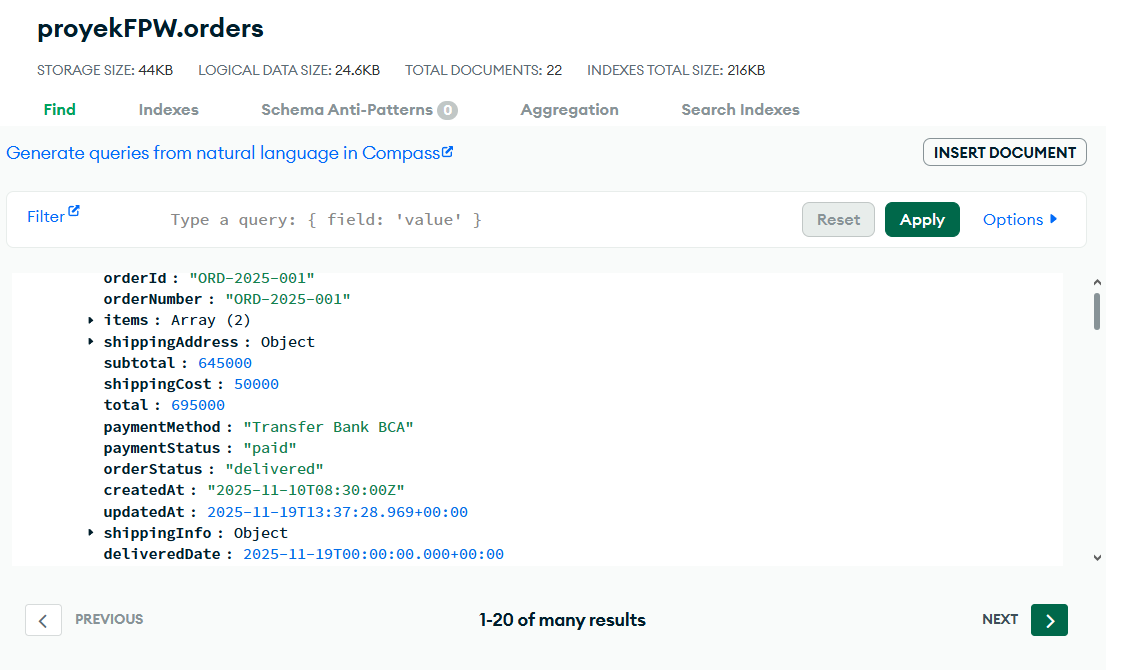
1. DEFINE protected procedure getUserOrders:
2. TRY:
3. CONNECT to database
4. FIND orders where userId matches authenticated user's ID
5. SORT orders by createdAt in descending order
6. RETURN orders as plain JavaScript objects
7. RETURN object containing the found orders
8. CATCH error:
9. LOG the error
10. THROW INTERNAL\_SERVER\_ERROR: 'Gagal mengambil data pesanan'

Pada Segmen Program 3.11, sistem akan memanggil procedure getUserOrders (02) dan di dalamnya sistem akan disambungkan dengan database (05). Kemudian sistem akan mencari order berdasarkan userId dan akan diurutkan berdasarkan createdAt secara descending dan kemudian akan dikembalikan order yang ditemukan (06 - 09). Tetapi jika terdapat error maka sistem akan mencatat error tersebut (10 - 12) dan memberikan pesan error “Gagal mengambil data pesanan” (13 - 18).

**Segmen Program 3.11 Contoh Program Melihat Status dan Riwayat Transaksi**

1. …
2. getUserOrders: protectedProcedure.query(async ({ ctx }) =>
3. {
4. try {
5. await connectDB();
6. const orders = await Order.find({
7. userId: ctx.user.id
8. }).sort({ createdAt: -1 }).lean();
9. return { orders };
10. } catch (error) {
11. console.error('[getUserOrders] Error:',
12. error);
13. throw new TRPCError({
14. code: 'INTERNAL\_SERVER\_ERROR',
15. message: 'Gagal mengambil data
16. pesanan',
17. cause: error,
18. });
19. }
20. }),
21. …

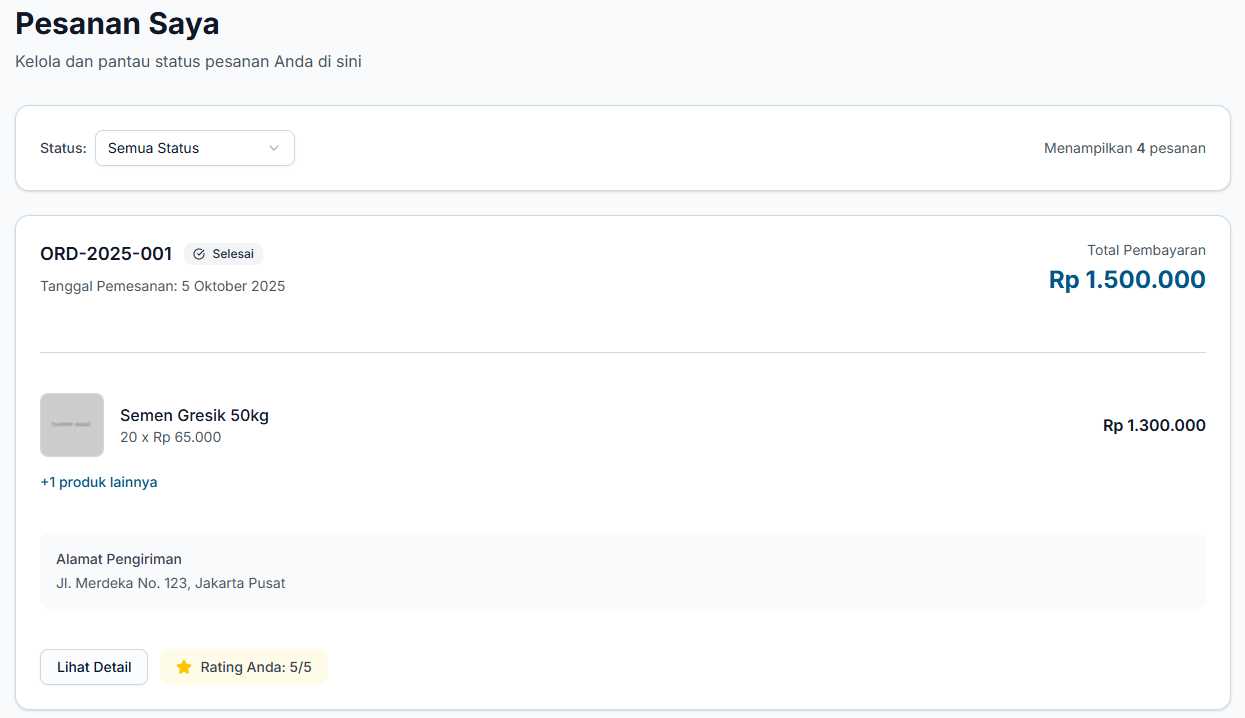
**Database yang Digunakan :**

****

**Gambar 3.13**

**Gambar Database Status Pesanan dan Riwayat Transaksi**

**Bentuk Gambar :**



**Gambar 3.14**

**Bentuk Gambar Status Pesanan dan Riwayat Transaksi**

**H. SP - 8 (Melihat Informasi Toko)**

Pada Algoritma 3.12, sistem memulai proses navigasi halaman informasi ketika Customer membuka halaman utama website (01) dan mengakses menu navigasi yang tersedia di footer (02). Selanjutnya, Customer memilih dan mengklik salah satu tautan informasi yang diinginkan, seperti "Tentang Kami", "Kontak Toko", "Syarat dan Ketentuan", atau "Kebijakan Privasi" (03 - 04). Sistem frontend kemudian mendeteksi adanya perubahan URL melalui mekanisme routing (05) dan mengeksekusi logika percabangan untuk menentukan konten yang tepat. Jika URL mengarah ke /about (06), sistem memanggil komponen AboutUsPage (07), sedangkan jika URL adalah /contact (09), sistem akan memanggil komponen ContactPage (10). Begitu pula jika URL sesuai dengan /terms (12), sistem memanggil komponen TermsPage (13), dan jika URL adalah /privacy (15) maka komponen PrivacyPage yang akan dipanggil (16). Setelah komponen yang sesuai teridentifikasi, browser me-render halaman tersebut (18) agar informasi dapat dibaca oleh pengguna, dan proses pun selesai (19).

**Algoritma 3.12 Contoh Algoritma Melihat Halaman Informasi Toko**

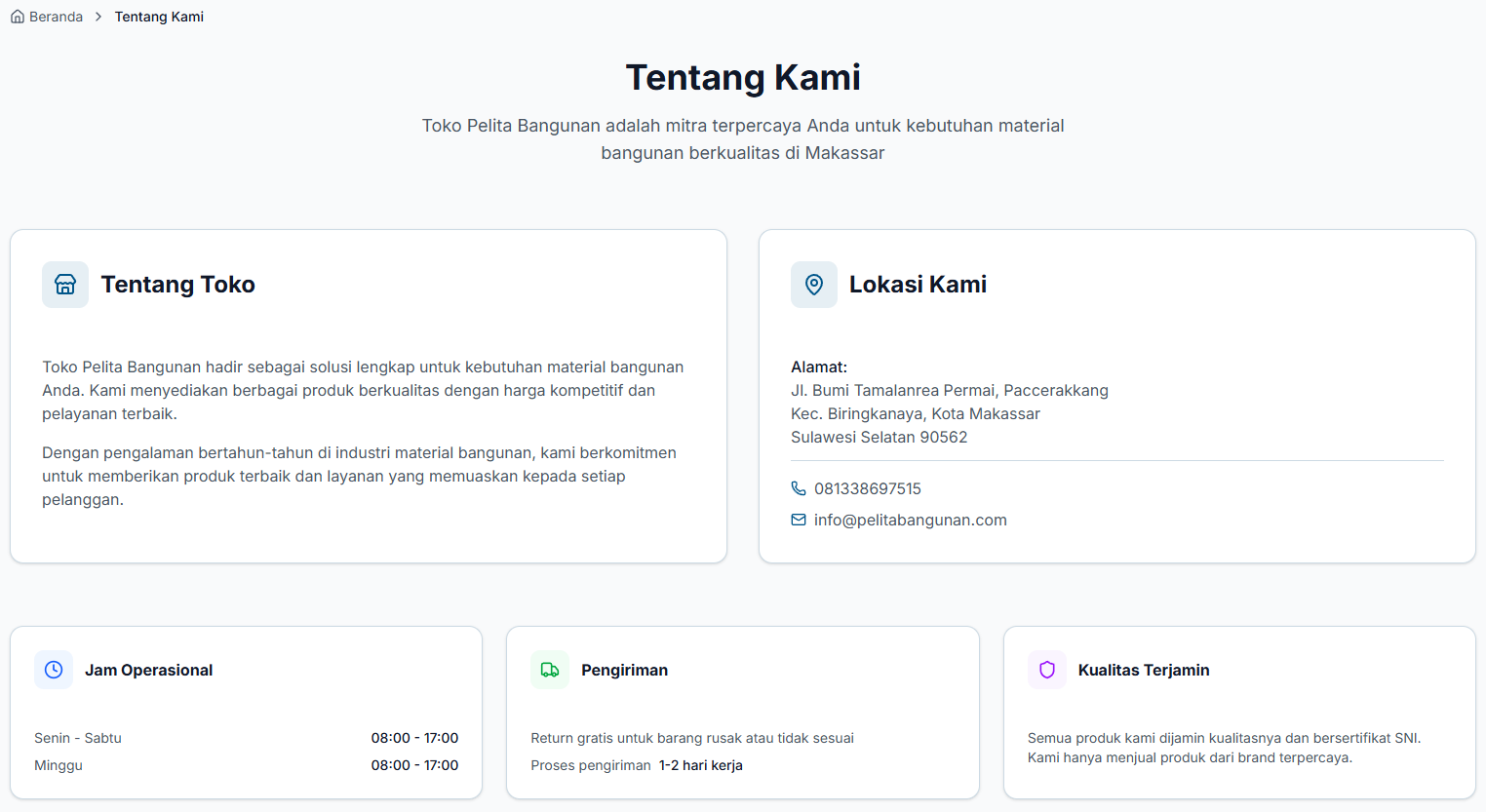
1. OPEN the main website
2. ACCESS the footer navigation menu
3. CHOOSE one of the information links “About Us” or “Kontak” or
4. “Syarat dan ketentuan” or “Kebijakan Privasi”
5. Frontend detects the URL change
6. IF URL = /about THEN
7. OPEN “About Us” page
8. END IF
9. IF URL = /contact THEN
10. OPEN “Kontak” page
11. END IF
12. IF URL = /terms THEN
13. OPEN “Syarat dan Ketentuan” page
14. END IF
15. IF URL = /privacy THEN
16. OPEN “Kebijakan Privacy” page
17. END IF
18. RENDER the requested page according to the loaded content
19. END

Pada Segmen Program 3.12, sistem akan menampilkan seluruh element - element yang bakal ditampilkan pada halaman website “About Us” beserta dengan styling dari masing - masing element, seperti div yang di dalamnya ada text “Tentang Kami”, paragraf yang berisi “Toko Pelita Bangunan adalah mitra terpercaya …”, dan element - element lainnya (02 - 10).

**Segmen Program 3.12 Contoh Program Melihat Halaman About Us**

1. …
2. <div classname=”text-center mb-16”>
3. <h1 classname=”text-4xl font-bold text-gray-900 mb-4”>
4. Tentang Kami
5. </h1>
6. <p classname=”text-lg text-gray-600 max-w-2xl mx-auto”>
7. Toko Pelita Bangunan adalah mitra terpercaya Anda untuk
8. kebutuhan material bangunan berkualitas di Makassar
9. </p>
10. </div>
11. …

**Bentuk Gambar :**

****

**Gambar 3.15**

**Bentuk Gambar Halaman About Us**

**I. SP - 9 (Melihat Informasi Bantuan)**

Pada Algoritma 3.13, sistem memulai proses navigasi halaman informasi ketika Customer membuka halaman utama website (01) dan mengakses menu navigasi yang tersedia di footer (02). Selanjutnya, Customer memilih dan mengklik salah satu tautan informasi yang diinginkan, seperti "Tentang Kami", "Kontak Toko", "Syarat dan Ketentuan", atau "Kebijakan Privasi" (03 - 04). Sistem frontend kemudian mendeteksi adanya perubahan URL melalui mekanisme routing (05) dan mengeksekusi logika percabangan untuk menentukan konten yang tepat. Jika URL mengarah ke /about (06), sistem memanggil komponen AboutUsPage (07), sedangkan jika URL adalah /contact (09), sistem akan memanggil komponen ContactPage (10). Begitu pula jika URL sesuai dengan /terms (12), sistem memanggil komponen TermsPage (13), dan jika URL adalah /privacy (15) maka komponen PrivacyPage yang akan dipanggil (16). Setelah komponen yang sesuai teridentifikasi, browser me-render halaman tersebut (18) agar informasi dapat dibaca oleh pengguna, dan proses pun selesai (19).

**Algoritma 3.13 Contoh Algoritma Melihat Halaman Informasi Bantuan**

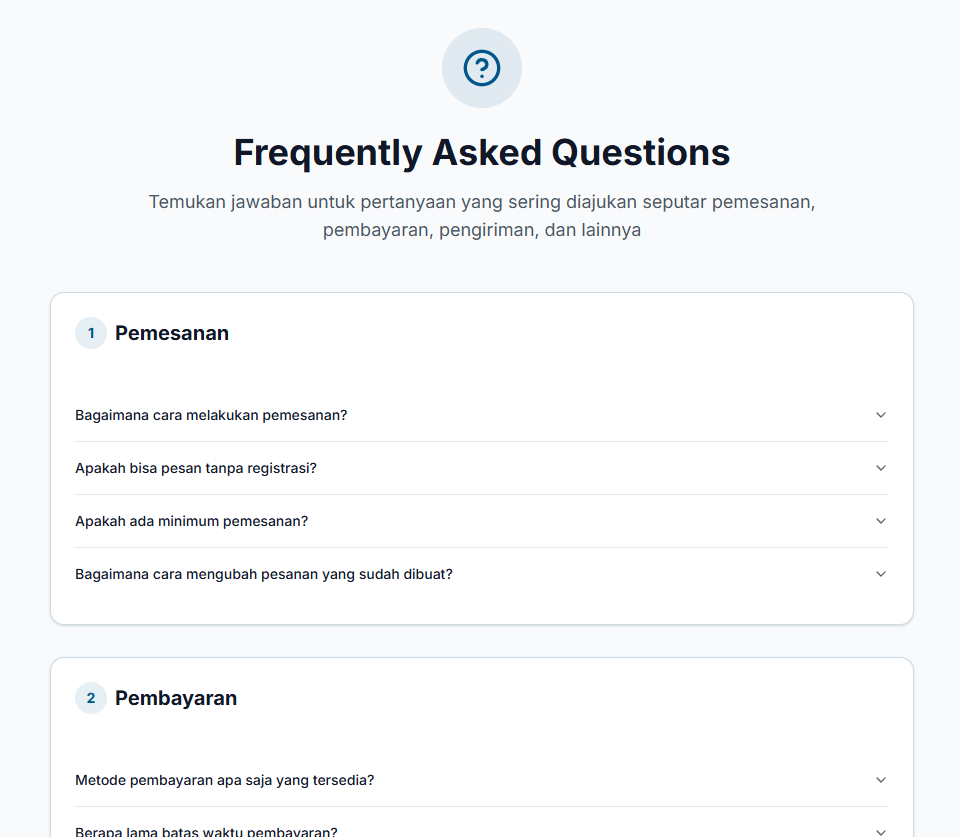
1. OPEN the main website
2. ACCESS the footer navigation menu
3. CHOOSE one of the information links “FAQ” or “Pengiriman” or
4. “Pengembalian” or “Pembayaran”
5. Frontend detects the URL change
6. IF URL = /faq THEN
7. OPEN “FAQ” page
8. END IF
9. IF URL = /shipping THEN
10. OPEN “Pengiriman” page
11. END IF
12. IF URL = /returns THEN
13. OPEN “Pengembalian” page
14. END IF
15. IF URL = /payment THEN
16. OPEN “Pembayaran” page
17. END IF
18. RENDER the requested page according to the loaded content
19. END

Pada Segmen Program 3.13, sistem akan menampilkan seluruh element – element dan komponen yang bakal ditampilkan pada halaman website “FAQ” beserta dengan styling dari masing - masing element dan komponen, seperti div yang di dalamnya dibungkus lagi dengan div yang kemudian diisi dengan komponen HelpCircle, kemudian ada text yang berisi “ Frequently Asked Question” dan element - element lainnya (02 - 14).

**Segmen Program 3.13 Contoh Program Melihat Halaman FAQ**

1. …
2. <div classname=”text-center mb-12”>
3. <div classname=”w-20 h-20 bg-primary/10 rounded-full flex
4. items-center justify-center mx-auto mb-6”>
5. <HelpCircle classname=”h-10 w-10 text-primary” />
6. </div>
7. <h1 classname=”text-4xl font-bold text-gray-900 mb-4”>
8. Frequently Asked Questions
9. </h1>
10. <p classname=”text-lg text-gray-600 max-w-2xl mx-auto”>
11. Temukan jawaban untuk pertanyaan yang sering diajukan
12. seputar pemesanan, pembayaran, pengiriman, dan lainnya
13. </p>
14. </div>
15. …

**Bentuk Gambar :**

****

**Gambar 3.16**

**Bentuk Gambar Halaman FAQ**

**J. SP - 10 (Fitur Notifikasi Untuk User Yang Sedang Login)**

Pada Algoritma 3.14, sistem memulai prosedur pengambilan notifikasi (01) dengan memasuki blok penanganan kesalahan atau try (02) dan sistem mengambil data dari koleksi Notification database (03). Sistem juga menghitung total dokumen yang cocok dengan kriteria (04) dan menentukan variabel hasMore untuk indikator halaman selanjutnya (05). Paket data yang berisi notifikasi, total, dan status hasMore kemudian dikembalikan sebagai respon (06). Apabila terjadi kegagalan selama proses (07), sistem akan menangkap kesalahan tersebut, mencatatnya ke dalam log konsol (08), dan melemparkan pesan Internal Server Error (09) sebelum mengakhiri prosedur (10).

**Algoritma 3.14 Contoh Algoritma Melihat Notifikasi User**

1. GET user notification:
2. TRY:
3. GET notifications FROM database
4. SET total = COUNT documents in notification
5. SET hasMore = (input.offset + input.limit < total)
6. RETURN {notifications, total, hasMore}
7. CATCH:
8. LOG the error
9. THROW INTERNAL\_SERVER\_ERROR: “Gagal mengambil data
10. Notification’
11. END

Pada Segmen Program 3.14, sistem akan mencari notifikasi dari database berdasarkan query yang kemudian akan disorting berdasarkan createdAt secara descending dan juga akan diskip berdasarkan input offset dan hasil yang ditampilkan akan di limit berdasarkan inputan limit dan kemudian dilean (02 - 06). Setelah mendapat notifications maka akan dihitung totalnya berdasarkan jumlah notification yang didapat berdasarkan query yang diberikan (07). Kemudian sistem akan mengembalikan object yang berisi notifications, total, dan juga hasMore yang dimana isinya adalah inputan offset ditambah inputan limit lebih kecil dari total (08 - 12).

**Segmen Program 3.14 Contoh Program Melihat Notifikasi User**

1. …
2. const notifications = await Notification.find(query)
3. .sort({ createAt: -1 })

**Segmen Program 3.14 (Lanjutan)**

1. .skip(input.offset)
2. .limit(input.limit)
3. .lean();
4. const total = await Notification.countDocuments(query);
5. return {
6. notifications,
7. total,
8. hasMore: input.offset + input.limit < total,
9. };
10. …

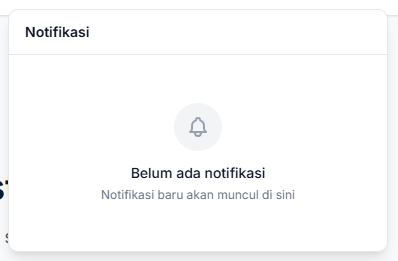
**Database yang digunakan :**

****

**Gambar 3.17**

**Gambar Database Fitur Notifikasi**

**Bentuk Gambar :**

****

**Gambar 3.18**

**Bentuk Gambar Fitur Notifikasi**

**K. SP - 11 (Mengatur Profile User Yang Sedang Login)**

Pada Algoritma 3.15, sistem memulai prosedur pembaruan profil pengguna dengan memvalidasi data masukan dari frontend, seperti nama lengkap dan nomor telepon (01). Setelah terhubung ke database, sistem menjalankan perintah update pada koleksi Pengguna, di mana pencarian data dilakukan berdasarkan ID pengguna (02). Jika data pengguna tidak ditemukan pada database (03) maka sistem akan mengirim pesan error “User not found” (04). Sistem mengganti data lama dengan data baru yang dikirimkan (06) dan akan menyimpan update tersebut (07) sebelum mengakhiri prosedur (08).

**Algoritma 3.15 Contoh Algoritma Mengubah Profile User**

1. GET fullname, phone from input
2. GET user from database byId
3. IF user not found THEN
4. THROW ERROR: “User not found”
5. END IF
6. UPDATE user fullname and phone
7. SAVE the update
8. END

Pada Algoritma 3.16, sistem memulai prosedur pembaruan password pengguna dengan memvalidasi data masukan dari frontend, seperti password lama dan password baru (01). Setelah terhubung ke database, sistem menjalankan perintah update pada koleksi Pengguna, di mana pencarian data dilakukan berdasarkan ID pengguna (02). Jika data pengguna tidak ditemukan pada database (03) maka sistem akan mengirim pesan error “User not found” (04). Kemudian sistem akan mengecek apakah inputan password lama yang diberikan cocok dengan password user (06). Jika password pengguna tidak cocok (07) maka sistem akan mengirim pesan error “Password not match” (08). Sebelum diupdate, inputan password baru akan dihash terlebih dahulu untuk menjaga keamanan user (10). Sistem mengganti password lama dengan password baru yang dikirimkan (11) dan akan menyimpan update tersebut (12) sebelum mengakhiri prosedur (13).

**Algoritma 3.16 Contoh Algoritma Mengubah Password User**

1. GET oldPassword, newPassword from input
2. GET user from database byId
3. IF user not found THEN
4. THROW ERROR: “User not found”
5. END IF
6. VERIFY oldPassword with currentPassword
7. IF password not match THEN
8. THROW ERROR: “Password not match”
9. END IF
10. HASH newPassword
11. UPDATE user password
12. SAVE the update
13. END

Pada Algoritma 3.17, sistem memulai prosedur pembaruan password pengguna dengan memvalidasi data masukan dari frontend, seperti lokasi alamat (01). Setelah terhubung ke database, sistem menjalankan perintah update pada koleksi Pengguna, di mana pencarian data dilakukan berdasarkan ID pengguna (02). Jika data pengguna tidak ditemukan pada database (03) maka sistem akan mengirim pesan error “User not found” (04). Kemudian sistem akan membuat id unik untuk inputan alamat (06). Jika alamat yang diisi adalah alamat pertama user (07) maka alamat tersebut akan dijadikan alamat default user (08). Sistem akan membuat object baru berdasarkan inputan alamat user (10), kemudian object tersebut akan ditambahkan ke data user (11) sebelum sistem mengakhiri prosedur (12).

**Algoritma 3.17 Contoh Algoritma Menambah Alamat User**

1. GET addressSchema from input
2. GET user from database byId
3. IF user not found THEN
4. THROW ERROR: “User not found”
5. END IF
6. GENERATE unique address id
7. IF this is the first address THEN
8. SET as default address
9. END IF
10. CREATE new address object
11. ADD address to user
12. END

Pada Segmen Program 3.15, sistem akan mencari user dari database berdasarkan userId (02). Jika user tidak ditemukan maka akan diberikan pesan error “User tidak ditemukan” (03 - 08), dan jika usernya sudah ditemukan maka fullname user yang lama akan diganti dengan fullname yang terbaru (09) dan begitu juga untuk phone akan digantikan dengan phone yang terbaru (10), kemudian perubahan tersebut akan disimpan oleh sistem (11). Kemudian sistem akan mengembalikan object yang berisi success, pesan berhasil, dan juga profile yang berisi fullname dan phone user yang baru (12 - 19).

**Segmen Program 3.15 Contoh Program Mengubah Profile User**

1. …
2. const user = await User.findById(ctx.user.id);
3. if (!user) {
4. throw new TRPCError({
5. code: 'NOT\_FOUND',
6. message: 'User tidak ditemukan',
7. });
8. }
9. user.fullName = input.fullName;
10. user.phone = input.phone;
11. await user.save();
12. return {
13. success: true,
14. message: 'Profil berhasil diperbarui',
15. profile: {
16. fullName: user.fullName,
17. phone: user.phone,
18. },
19. };
20. …

Pada Segmen Program 3.16, sistem akan mencari user dari database berdasarkan userId (02). Jika user tidak ditemukan maka akan diberikan pesan error “User tidak ditemukan” (03 - 08). Kemudian passwordnya juga akan dicek apakah inputan oldPassword mirip dengan user password saat ini (09 - 10), jika tidak valid maka akan diberikan pesan error “Password lama tidak sesuai” (11 - 16). Jika valid maka inputan newPassword akan dihash untuk memberikan keamanan tambahan (17) dan password dari user akan diubah dengan password yang sudah dihash (18), serta sistem akan menyimpan perubahan password tersebut (19). Terakhir, sistem akan mengembalikan object dimana isinya berupa success, dan pesan “Password berhasil diubah” (20 - 23).

**Segmen Program 3.16 Contoh Program Mengubah Password User**

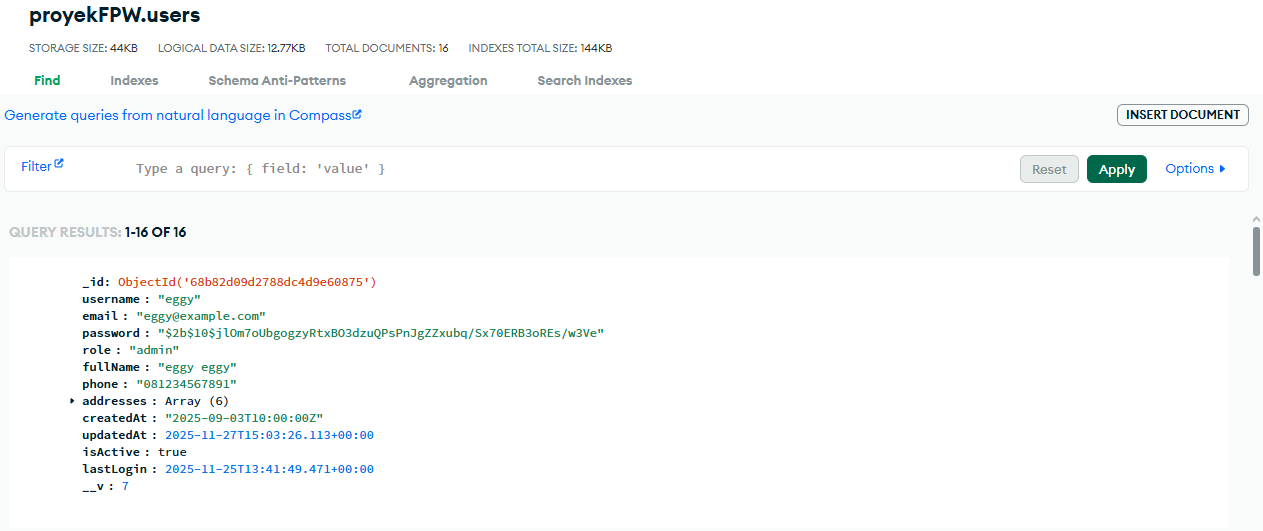
1. …
2. const user = await User.findById(ctx.user.id);
3. if (!user) {
4. throw new TRPCError({
5. code: 'NOT\_FOUND',
6. message: 'User tidak ditemukan',
7. });
8. }
9. const isPasswordValid = await compare(input.oldPassword,
10. user.password);
11. if (!isPasswordValid) {
12. throw new TRPCError({
13. code: 'UNAUTHORIZED',
14. message: 'Password lama tidak sesuai',
15. });
16. }
17. const hashedPassword = await hash(input.newPassword, 10);
18. user.password = hashedPassword;
19. await user.save();
20. return {
21. success: true,
22. message: 'Password berhasil diubah',
23. };
24. …

Pada Segmen Program 3.17, sistem akan membuat addressId berdasarkan userId dan waktu saat ini (02), dan membuat variabel isDefault untuk pengecekan apakah sudah ada alamat yang dimiliki oleh user (03). Kemudian sistem akan membuat alamat baru dimana isinya berupa id, label, recipientName, phoneNumber, dan attribut lainnya (04 - 18). Setelah dibuatkan alamat baru, alamat tersebut akan dimasukkan kedalam alamat user (19) dan sistem akan menyimpan perubahan tersebut (20). Terakhir, sistem akan mengembalikan object yang berisi success, pesan “Alamat berhasil ditambahkan”, dan juga alamat yang baru dibuat (21 - 25).

**Segmen Program 3.17 Contoh Program Menambah Alamat User**

1. …
2. const addressId = `addr-${ctx.user.id}-${Date.now()}`;
3. const isDefault = user.addresses.length === 0;
4. const newAddress = {
5. id: addressId,
6. label: input.label,
7. recipientName: input.recipientName,
8. phoneNumber: input.phoneNumber,
9. fullAddress: input.fullAddress,
10. district: input.district,
11. city: input.city,
12. province: input.province,
13. postalCode: input.postalCode,
14. notes: input.notes || '',
15. isDefault,
16. latitude: input.latitude,
17. longitude: input.longitude,
18. };
19. user.addresses.push(newAddress);
20. await user.save();
21. return {
22. success: true,
23. message: 'Alamat berhasil ditambahkan',
24. address: newAddress,
25. };
26. …

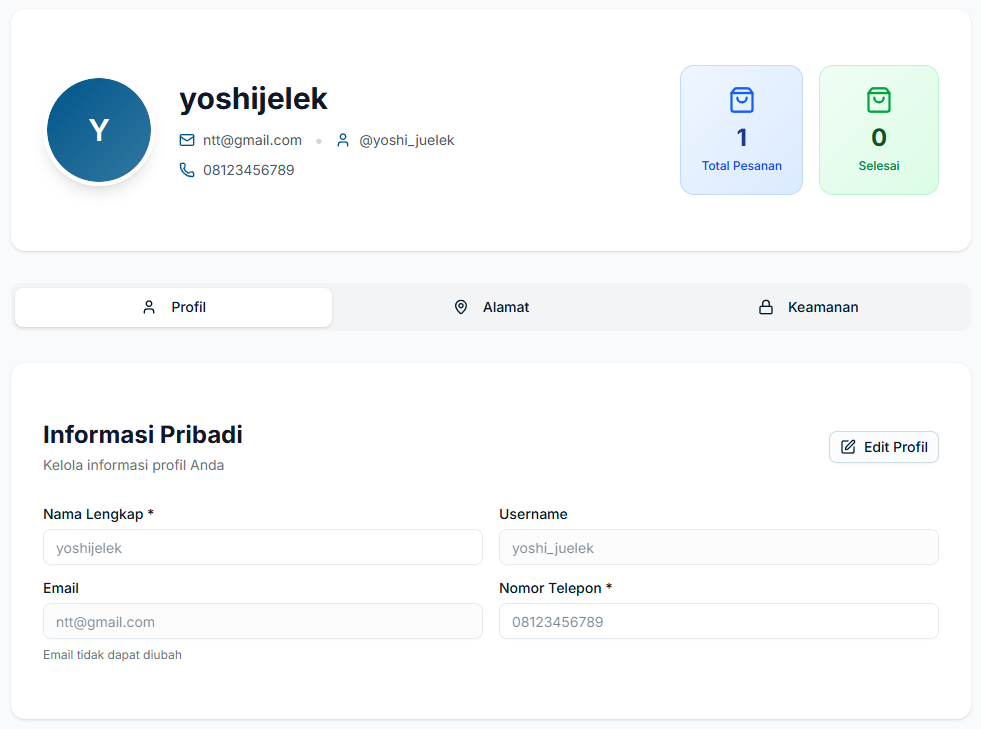
**Database yang digunakan :**

****

**Gambar 3.19**

**Gambar Database Profile User**

**Bentuk Gambar :**

****

**Gambar 3.20**

**Bentuk Gambar Mengatur Profile User**

**3.5 Sprint Review**

Pada sesi ini, tim pengembang melakukan demonstrasi tentang bagaimana alur penggunaan program yang dibuat kepada pemiliki UD. Pelita Bangunan. Sesi ini bertujuan untuk mengumpulkan hasil kerja dan juga mendapatkan kritik/saran langsung untuk perkembangan ke sprint berikutnya.

Secara keseluruhan, pemilik toko memberikan tanggapan yang sangat positif. Beliau memberitahu bahwa fungsionalitas dasar seperti alur registrasi customer, proses login, tampilan katalog produk, dan tampilan keranjang sudah berjalan dengan baik dan sesuai dengan apa yang diharapkan.



**Gambar 3.21**

**Bukti Hasil Review Bersama Pemilik UD.**

Selain apresiasi tersebut, tetap ada masukan yang diberikan oleh pemilik toko. Masukan ini berfokus pada halaman inventori dibagian sisi Admin, agar tampilan nya lebih gampang dipahami oleh user ketika pertama kali membuka halaman tersebut. Beliau meminta agar urutan kolom pada table inventori disesuaikan. Secara spesifik, beliau menyarankan agar kolom “Source” dan “Tipe” digabungkan saja menjadi 1 kolom menjadi kolom “Keterangan”, karena setelah dilihat kedua kolom tersebut memiliki isi yang hampir mirip satu sama lain. Dan juga beliau ingin untuk urutan kolom baru nya menjadi “Nama Produk” – “Stok Awal” – “Keterangan” – “Kuantiti” – “Stok Akhir”. Serta meminta agar judul kolomnya konsisten saja menggunakan Bahasa Indonesia.

**Tabel 3.3**

**Tabel Hasil Sprint Review**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Product Backlog ID** | **Deskripsi** | **Status** |
| SR - 1 | PB - 1 | Masukan Data ke Database | Selesai |

**Tabel 3.3**

**(Lanjutan)**

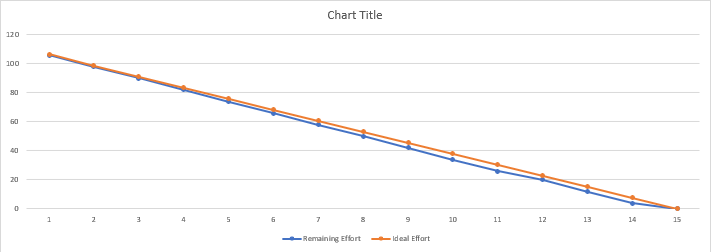
|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Product Backlog ID** | **Deskripsi** | **Status** |
| SR - 2 | PB - 2 | Mendaftar serta login ke akun yang dibuat agar bisa mengakses fitur pembelian dan melacak pesanan. | Selesai |
| SR - 3 | PB - 3 | Melihat katalog produk lengkap dengan detail dari barang yang dipilih agarmendapat informasi lebih mengenai produk yang ingin dibeli. | Selesai |
| SR - 4 | PB - 4 | Fitur pencarian barang berdasarkan nama produk agar proses pencarian barang menjadi lebih efisien. | Selesai |
| SR - 5 | PB - 5 | Menambah dan mengatur produk di cart agar dapat mengumpulkan barang yang ingin dibeli sebelum melakukan pembayaran. | Selesai |
| SR - 6 | PB - 6 | Melakukan checkout dan pembayaran agar dapat menyelesaikan transaksi pembelian. | Selesai |
| SR - 7 | PB - 7 | Melihat status pesanan dan riwayat transaksi agar bisa memantau progres produk pembelian. | Selesai |
| SR - 8 | PB - 8 | Melihat informasi toko seperti About Us, Kontak, Syarat dan Ketentuan dan lain - lain agar customer bisa tahu informasi mengenai toko. | Selesai |
| SR - 9 | PB - 9 | Melihat informasi bantuan seperti FAQ, Pengiriman, dan lain - lain agar customer mengetahui sistem kerja toko. | Selesai |
| SR - 10 | PB - 10 | Fitur notifikasi agar customer selalu mendapat informasi terbaru mengenai produk jualan atau pesanan customer. | Selesai |
| SR - 11 | PB - 11 | Mengatur profile agar dapat mengelola informasi pribadi seperti alamat, password, dan lain - lain. | Selesai |

Umpan balik ini diterima oleh tim sebagai masukkan yang bagus. Permintaan penyesuaian tampilan tabel inventori ini dicatat dan akan diprioritaskan untuk diimplementasikan pada Scrum 2 untuk mengembangkan kenyamanan penggunaan di dashboard Admin.

**3.6 Burndown Chart**

Burndown Chart digunakan untuk melacak kemajuan kerja tim terhadap waktu. Grafik Burndown bertujuan memvisualisasikan jumlah pekerjaan yang tersisa (dalam jam atau poin tugas) dari hari ke hari selama Sprint berlangsung. Detail lebih lanjut mengenai tabel burndown chart dapat dilihat pada lampiran A.

Tabel burndown chart berfungsi melacak kemajuan kerja tim, tabel tersebut memecah setiap tugas dari Sprint Backlog dan melacak jam kerja actual yang dihabiskan oleh tim pengembang dalam mengembangkan program yang dibuat. Untuk total waktu yang di butuhkan untuk mengembangkan program adalah 114 jam.

****

**Gambar 3.22**

**Grafik Burndown Chart Sprint 1**

Contoh grafik burndown di atas menunjukkan Ideal Line (garis Orange) yang merepresentasikan laju kerja yang ideal, dan Actual Line (garis biru) yang menunjukkan sisa pekerjaan aktual.

**BAB IV**

**SCRUM II**

Pada bab ini, menjelaskan mengenai pelaksanaan siklus kerja Scrum kedua, atau Sprint 2. Fokus utama dari Sprint 2 adalah untuk menyempurnakan sistem yang telah dibangun pada Sprint 1 dengan memperbaiki fitur-fitur yang sekiranya kurang maksimal pada Sprint 1. Selain itu, sprint ini juga berfokus pada peningkatan performa sistem dan antarmuka pengguna agar pengalaman penggunaan menjadi lebih optimal.

#### **4.1 Product Backlog**

Product Backlog untuk Sprint 2 akan berfokus pada fitur-fitur yang ingin direvisi pada Sprint 1, terutama yang berkaitan dengan tampilan pada halaman dashboard admin. Tabel di bawah ini akan memberikan penjelasan lebih jelas mengenai fitur-fitur apa saja yang dipilih untuk dikerjakan pada Sprint 2:

**Tabel 4.1**

**Tabel Product Backlog**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Deskripsi** | **Aktor** | **Prioritas** | **Sprint** | **Waktu** |
| PB - 1 | Masukan Data ke Database | Admin | 1 | 1 | 6 Jam |
| PB - 2 | Mendaftar serta login ke akun yang dibuat agar bisa mengakses fitur pembelian dan melacak pesanan. | Customer | 1 | 1 | 26 Jam |
| PB - 3 | Melihat katalog produk lengkap dengan detail dari barang yang dipilih agarmendapat informasi lebih mengenai produk yang ingin dibeli. | Customer | 2 | 1 | 8 Jam |
| PB - 4 | Fitur pencarian barang berdasarkan nama produk agar proses pencarian barang menjadi lebih efisien. | Customer | 2 | 1 | 6 Jam |

**Tabel 4.1**

**(Lanjutan)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Deskripsi** | **Aktor** | **Prioritas** | **Sprint** | **Waktu** |
| PB - 5 | Menambah dan mengatur produk di cart agar dapat mengumpulkan barang yang ingin dibeli sebelum melakukan pembayaran. | Customer | 1 | 1 | 10 Jam |
| PB - 6 | Melakukan checkout dan pembayaran agar dapat menyelesaikan transaksi pembelian. | Customer | 1 | 1 | 12 Jam |
| PB - 7 | Melihat status pesanan dan riwayat transaksi agar bisa memantau progres produk pembelian. | Customer | 2 | 1 | 10 Jam |
| PB - 8 | Melihat informasi toko seperti About Us, Kontak, Syarat dan Ketentuan, dan lain - lain agar customer bisa tahu informasi mengenai toko. | Customer | 3 | 1 | 8 Jam |
| PB - 9 | Melihat informasi bantuan seperti FAQ, Pengiriman, dan lain - lain agar customer mengetahui sistem kerja toko. | Customer | 3 | 1 | 8 Jam |
| PB - 10 | Fitur notifikasi agar customer selalu mendapat informasi terbaru mengenai produk jualan atau pesanan customer. | Customer | 1 | 1 | 8 Jam |
| PB - 11 | Mengatur profile agar dapat mengelola informasi pribadi seperti alamat, password, dan lain - lain. | Customer | 1 | 1 | 12 Jam |
| PB - 12 | Mengelola stok produk secara real-time agar stok barang yang ditampilkan ke customer selalu akurat. | Staff | 1 | 2 | 14 Jam |

**Tabel 4.1**

**(Lanjutan)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Deskripsi** | **Aktor** | **Prioritas** | **Sprint** | **Waktu** |
| PB - 13 | Memproses permintaan pengembalian produk agar proses retur barang dapat tercatat dan diselesaikan sesuai prosedur. | Staff | 3 | 2 | 12 Jam |
| PB - 14 | Menginput nomor resi pengiriman agar customer dapat melacak status pengiriman mereka. | Staff | 2 | 2 | 6 Jam |
| PB - 15 | Menambah dan mengelola data produk agar katalog produk yang ditampilkan kepada customer selalu akurat. | Admin | 1 | 2 | 12 Jam |
| PB - 16 | Melihat laporan transaksi, informasi inventori, dan data pengguna agar dapat membuat keputusan bisnis yang terstruktur. | Admin | 2 | 2 | 12 Jam |
| PB - 17 | Menambah dan mengelola data voucher. | Admin | 2 | 2 | 8 Jam |
| PB - 18 | Menambahkan, mengelola, dan mengaktifasikan akun staff dan admin. | Admin | 1 | 2 | 12 Jam |
| PB - 19 | Menambah dan mengelola kategori produk dan unit konversi. | Admin | 1 | 2 | 14 Jam |

**4.2 Sprint Backlog**

Setelah Product Backlog untuk Sprint 2 dipilih, tim pengembang akan memecahnya menjadi tugas-tugas teknis yang lebih kecil dan terstruktur. Ini akan menghasilkan Sprint Backlog baru, yang menjadi rencana kerja harian bagi tim pengemban. Tim pengembang dapat menambah, mengubah, atau menghapus tugas sesuai kebutuhan. Tabel di bawah ini akan memberikan penjelasan lebih jelas mengenai Sprint Backlog dari sistem ini untuk Sprint 2:

**Tabel 4.2**

**Tabel Sprint Backlog**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Product Backlog** | **Deskripsi** | **Tugas** |
| SP - 1 | PB - 12 | (SP-1.1) Mengupdate API checkout agar otomatis mengurangi stok produk saat pesanan berhasil. | Given |
| (SP-1.2) Buat API dan Halaman UI untuk mengelola stok produk. | Aaron |
| SP - 2 | PB - 13 | (SP-2.1) Buat API endpoint untuk pengembalian produk. | Christopani |
| (SP-2.2) Buat Halaman UI "Form Pengembalian" (Customer) dan "Manajemen Pengembalian" (Staff). | Aaron |
| SP - 3 | PB - 14 | (SP-3.1) Mengupdate API pesanan agar Staff dapat menambahkan nomor resi. | Given |
| (SP-3.2) Buat UI input nomor resi pada halaman "Manajemen Pesanan" di dasboard Staff. | Aaron |
| SP - 4 | PB - 15 | (SP-4.1) Buat API CRUD dasar untuk data produk. | Christopani |
| (SP-4.2) Buat halaman UI form untuk tambah/edit produk. | Aaron |
| SP - 5 | PB - 16 | (SP-5.1) Buat API endpoint untuk data 10 laporan Admin. | Egbert |
| (SP-5.2) Buat halaman UI laporan (tabel & grafik) di dasboard Admin. | Aaron |
| SP - 6 | PB - 17 | (SP-6.1) Buat API endpoint untuk menerima input data voucher baru. | Christopani |
| (SP-6.2) Buat API endpoint untuk mengambil semua list voucher. | Christopani |
| (SP-6.3) Buat halaman UI Manajemen Voucher di dashboard admin. | Aaron |
| (SP-6.4) Implementasi form untuk penambahan voucher baru. | Aaron |
| SP - 7 | PB - 18 | (SP-7.1) Buat API endpoint untuk membuat akun staff atau admin baru. | Christopani |
| (SP-7.2) Buat API endpoint untuk mengambil semua list user yang memiliki role staff dan admin. | Christopani |
| (SP-7.3) Buat API endpoint untuk mengubah status dari staff tertentu. | Given |

**Tabel 4.2**

**(Lanjutan)**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Product Backlog** | **Deskripsi** | **Tugas** |
| SP - 7 | PB - 18 | (SP-7.4) Buat API endpoint untuk mengubah data dari staff tertentu. | Given |
| (SP-7.5) Buat halaman UI Manajemen Pegawai di dashboard admin. | Aaron |
| (SP-7.6) Implementasi form tambah atau edit staff. | Aaron |
| SP - 8 | PB - 19 | (SP-8.1) Buat API CRUD untuk kategori. | Christopani |
| (SP-8.2) Buat API CRUD untuk unit. | Given |
| (SP-8.3) Buat halaman UI produk di dashboard admin. | Aaron |
| (SP-8.4) Implementasi tabel dan form untuk kategori dan unit. | Aaron |

**4.3 Pembuktian Sprint**

Pada sub-bab ini, akan diberikan bukti implementasi dari setiap Product Backlog yang telah dibuat. Pembuktian disajikan dalam tiga bentuk, yaitu: alur logika (algoritma), segmen kode program, database yang terhubung dan bentuk gambar dari UI sistem yang telah berhasil dibuat.

**A. SP - 1 (Mengelola Stok Produk)**

Pada Algoritma 4.1, sistem memulai proses dengan menghitung jumlah total dokumen StockMovement yang memenuhi kriteria filter (01) untuk mengetahui banyaknya data yang tersedia secara keseluruhan. Selanjutnya, sistem melakukan pencarian dokumen StockMovement berdasarkan filter yang sama (02). Agar data tampil terurut dari yang paling baru, sistem mengurutkan hasil pencarian berdasarkan waktu pembuatan secara menurun (03). Proses pengambilan data ini dibatasi jumlahnya sesuai nilai limit yang diminta (04) dan melewati sejumlah data tertentu berdasarkan nilai offset (05) untuk keperluan pemisahan halaman (pagination). Setelah data diperoleh, sistem mengembalikannya dalam format objek JavaScript biasa (06). Terakhir, sistem menyusun dan mengembalikan hasil akhir dalam bentuk objek (07) yang memuat daftar pergerakan stok yang ditemukan (08), jumlah total data keseluruhan (09), serta indikator hasMore (10 - 11) yang dihitung dengan membandingkan posisi data saat ini dengan total data untuk menentukan apakah masih ada halaman berikutnya yang tersedia.

**Algoritma 4.1 Contoh Algoritma Mengelola Stok Produk**

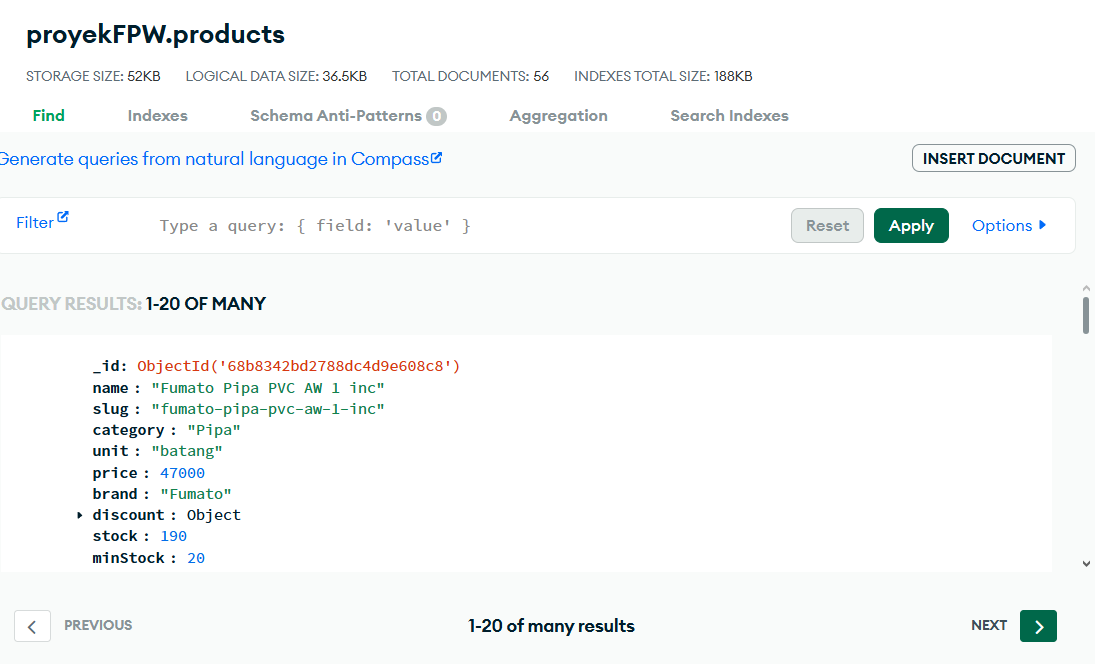
1. COUNT total documents in StockMovement matching the filter
2. FIND StockMovement documents matching the filter:
3. SORT by createdAt in descending order
4. LIMIT by input.limit
5. SKIP by input.offset
6. RETURN as plain JavaScript objects
7. RETURN object containing:
8. movements (the found documents)
9. total (total count)
10. hasMore (boolean indicating if more results exist, calculated
11. by comparing current offset + limit with total)

Pada Segmen Program 4.1, sistem akan menghitung total stock produk dari database (02) dan juga mencari movement dari database berdasarkan filter, dan hasilnya akan disorting berdasarkan createdAt secara descending, dilimit berdasarkan inputan limit, diskip berdasarkan inputan offset, dan akan dilean (03 - 07). Kemudian sistem akan mengembalikan object yang berisi movement, total, dan juga hasMore yang berisi inputan offset ditambah dengan inputan limit yang lebih kecil dari total (08 - 12).

**Segmen Program 4.1 Contoh Program Mengelola Stok Produk**

1. …
2. const total = await StockMovement.countDocuments(filter);
3. const movements = await StockMovement.find(filter)
4. .sort({ createdAt: -1 })
5. .limit(input.limit)
6. .skip(input.offset)
7. .lean();
8. return {
9. movements,
10. total,
11. hasMore: input.offset + input.limit < total,
12. };
13. …

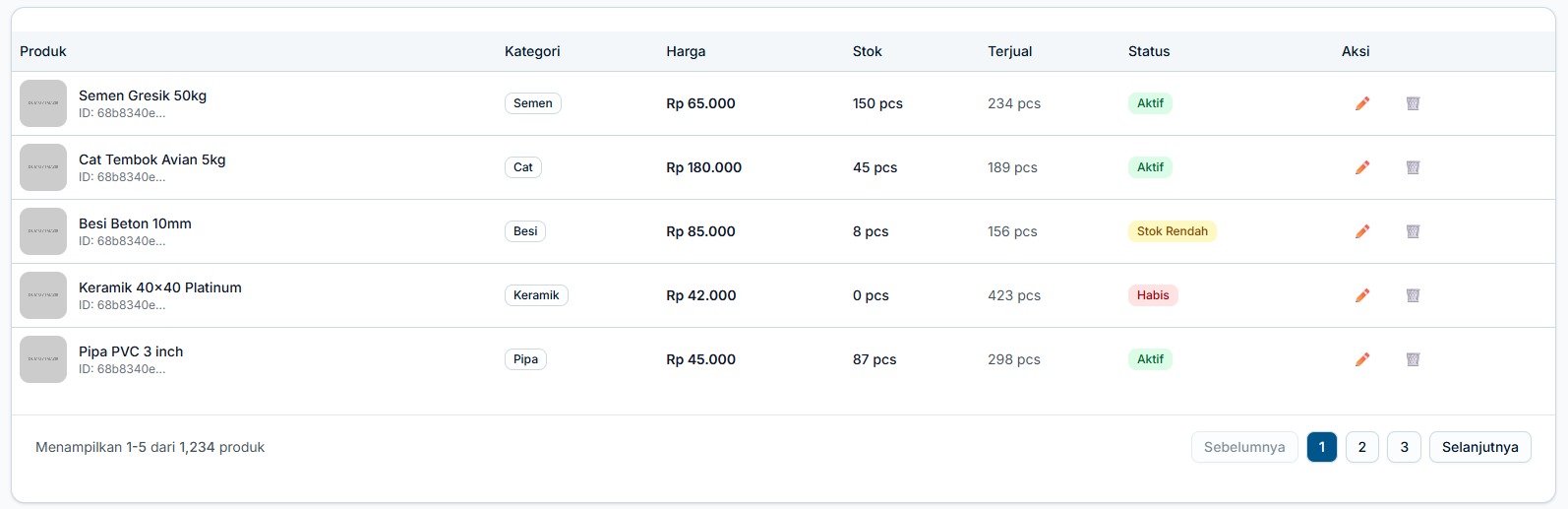
**Database yang Digunakan :**



**Gambar 4.1**

**Gambar Database Stok Produk**

**Bentuk Gambar :**



**Gambar 4.2**

**Bentuk Gambar Mengelola Stok Produk**

**B. SP - 2 (Memproses Permintaan Pengembalian Produk)**

Pada Algoritma 4.2, sistem memulai proses dengan mencari data permintaan pengembalian (return request) menggunakan returnId yang diinputkan (01). Jika data permintaan tersebut tidak ditemukan (02), sistem akan menghentikan operasi dan melemparkan pesan kesalahan "Return request tidak ditemukan" (03). Selanjutnya, jika permintaan ditemukan, sistem memeriksa statusnya saat ini; apabila statusnya bukan 'pending' (05), sistem menolak proses dengan pesan kesalahan "Return request sudah diproses" (06). Jika status valid, sistem mengambil waktu saat ini dalam format ISO (08) lalu memperbarui status permintaan menjadi 'approved' (09), sekaligus menetapkan tanggal persetujuan (10) dan waktu pembaruan terakhir (11). Setelah menyimpan perubahan pada permintaan pengembalian tersebut (12), sistem mencari pesanan utama yang terkait dengan ID pesanan tersebut (13). Apabila pesanan ditemukan (14), sistem memperbarui status pengembalian pada pesanan menjadi 'approved' (15) dan menyimpan perubahan tersebut (16). Proses diakhiri dengan sistem mengembalikan status sukses beserta pesan konfirmasi bahwa "Return request disetujui" (18).

**Algoritma 4.2 Contoh Algoritma Permintaan Pengembalian Produk**

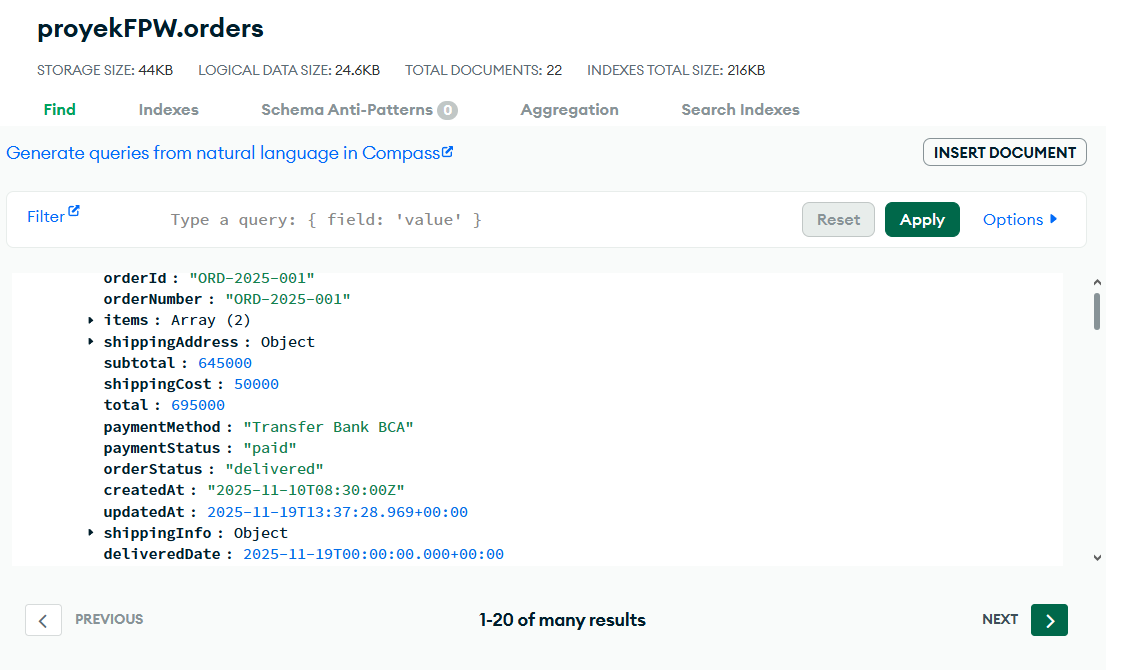
1. FIND return request by input.returnId
2. IF return request not found THEN
3. THROW NOT\_FOUND error: 'Return request tidak ditemukan'
4. END IF
5. IF return request status is not 'pending' THEN
6. THROW BAD\_REQUEST error: 'Return request sudah diproses'
7. END IF
8. SET now to current ISO date and time
9. SET return request status to 'approved'
10. SET return request approvedDate to now
11. SET return request updatedAt to now
12. SAVE return request changes
13. FIND order by return request's orderId
14. IF order found THEN
15. SET order returnStatus to 'approved'
16. SAVE order changes
17. END IF
18. RETURN success: true and message: 'Return request disetujui'

Pada Segmen Program 4.2, sistem akan mencari returnRequest dari database berdasarkan returnId (02), jika tidak ditemukan didatabase maka sistem akan memberikan pesan error “Return request tidak ditemukan” (03 - 08) dan jika returnRequest ditemukan tetapi statusnya bukan pending maka sistem juga akan memberikan pesan error “Return request sudah diproses” (09 - 14). Kemudian jika semua proses tersebut terlewati denga naman maka sistem akan menyimpan waktu saat ini kedalam variabel “now” (15) dan status dari request akan di approved, serta approveDate dan updatedAt akan diisi dengan variabel “now” yang barusan dibuat (16 - 18) dan sistem akan menyimpan returnRequest tersebut (19). Setelah itu sistem akan mencari order yang ingin diretur dari database berdasarkan orderId (20 - 22), dan jika ordernya ditemukan maka status retur dari order tersebut berubah menjadi approved dan sistem akan menyimpan perubahan tersebut (23 - 26). Terakhir, sistem akan mengembalikan object yang berisi success, dan pesan “Return request disetujui” (27 - 29).

**Segmen Program 4.2 Contoh Program Permintaan Pengembalian Produk**

1. …
2. const returnRequest = await Return.findById(input.returnId);
3. if (!returnRequest) {
4. throw new TRPCError({
5. code: 'NOT\_FOUND',
6. message: 'Return request tidak ditemukan',
7. });
8. }
9. if (returnRequest.status !== 'pending') {
10. throw new TRPCError({
11. code: 'BAD\_REQUEST',
12. message: 'Return request sudah diproses',
13. });
14. }
15. const now = new Date().toISOString();
16. returnRequest.status = 'approved';
17. returnRequest.approvedDate = now;
18. returnRequest.updatedAt = now;
19. await returnRequest.save();
20. const order = await Order.findById(
21. returnRequest.orderId
22. );
23. if (order) {
24. order.returnStatus = 'approved';
25. await order.save();
26. }
27. return {
28. success: true, message: 'Return request disetujui'
29. };

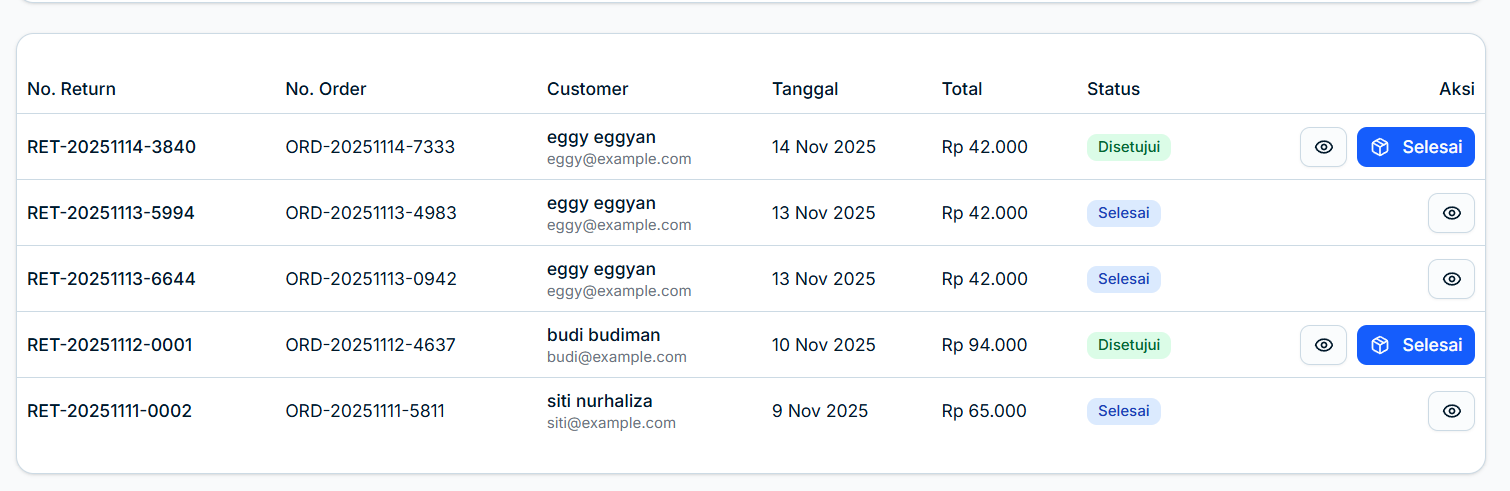
**Database yang Digunakan :**

****

**Gambar 4.3**

**Gambar Database Pengembalian Produk**

**Bentuk Gambar :**



**Gambar 4.4**

**Bentuk Gambar Proses Permintaan Pengembalian Produk**

**C. SP - 3 (Menginput Nomor Resi Pengiriman)**

Pada Algoritma 4.3, sistem memulai dengan membuat objek query dasar yang berisi orderId dari input (01). Selanjutnya, sistem melakukan pemeriksaan hak akses; jika peran pengguna yang sedang login bukan 'admin' atau 'staff' (02), sistem secara otomatis menambahkan userId pengguna tersebut ke dalam kriteria query (03) untuk membatasi akses sehingga pengguna hanya dapat melihat pesanannya sendiri. Setelah kriteria pencarian ditetapkan, sistem mencari satu dokumen pesanan yang cocok dengan query tersebut (05). Dalam proses pencarian ini, sistem juga melengkapi (populate) kolom 'userId' dengan detail informasi seperti 'fullName', 'name', 'email', dan 'phone' (06 - 07), lalu mengambil data tersebut sebagai objek JavaScript biasa (08). Jika pesanan tidak ditemukan berdasarkan kriteria yang ada (09), sistem akan menghentikan proses dan melemparkan error "Order not found" (10). Namun, jika data berhasil ditemukan, sistem akan mengembalikan objek yang berisi detail pesanan tersebut (12) kepada peminta.

**Algoritma 4.3 Contoh Algoritma Menginput Nomor Resi Pengiriman**

1. CREATE query object with orderId from input
2. IF authenticated user's role is NOT 'admin' or 'staff' THEN
3. ADD userId from authenticated user to query
4. END IF
5. FIND one order matching the query:
6. POPULATE 'userId' field with 'fullName', 'name', 'email',
7. ‘phone’
8. RETURN as plain JavaScript object
9. IF order not found THEN
10. THROW NOT\_FOUND error: 'Order not found'
11. END IF
12. RETURN object containing the found order

Pada Segmen Program 4.3, sistem akan mencari query orderId berdasarkan inputan orderId (02 - 03). Jika user bukan seorang admin atau staff, sistem akan menambahkan batasan userId ke dalam query pencarian (04 - 06). Kemudian sistem akan mencari order dalam database berdasarkan query dan akan dilean (07 - 09). Jika order tidak ditemukan maka sistem akan memberikan pesan error “Order not found” (10 - 15). Terakhir sistem akan mengembalikan order yang sudah ditemukan (16).

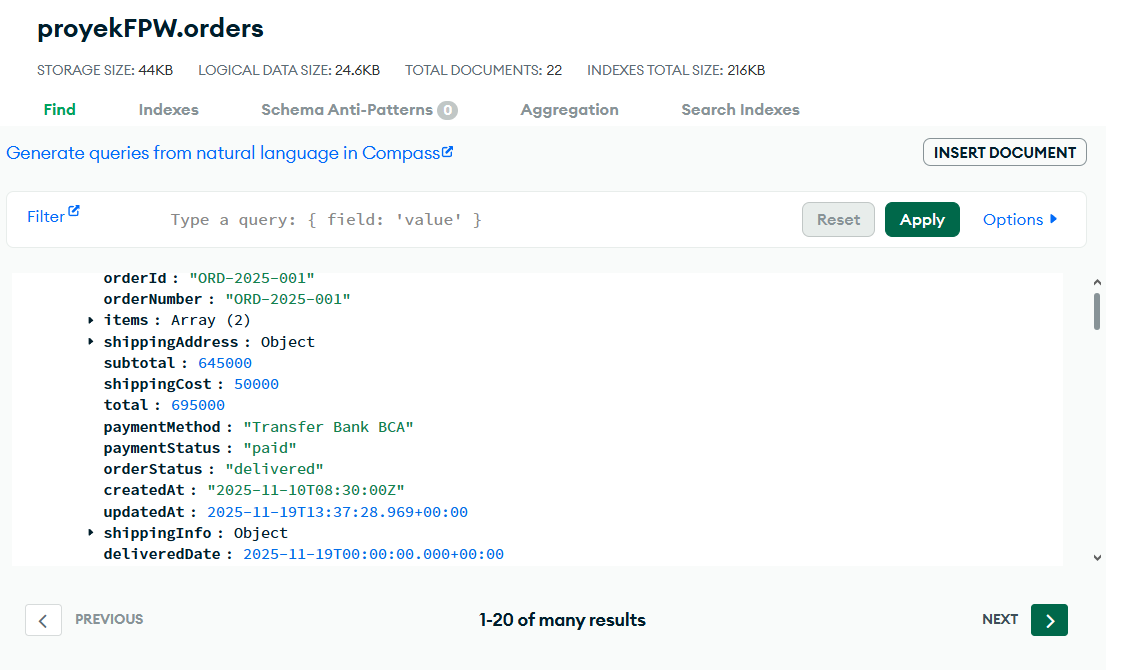
**Segmen Program 4.3 Contoh Program Menginput Nomor Resi Pengiriman**

1. …
2. const query: { orderId: string; userId?: string } = {
3. orderId: input.orderId };
4. if (!['admin', 'staff'].includes(ctx.user.role)) {
5. query.userId = ctx.user.id;
6. }
7. const order = await Order.findOne(query)
8. .populate('userId', 'fullName name email phone')

**Segmen Program 4.3 (Lanjutan)**

1. .lean();
2. if (!order) {
3. throw new TRPCError({
4. code: 'NOT\_FOUND',
5. message: 'Order not found',
6. });
7. }
8. return { order };
9. …

**Database yang Digunakan :**

****

**Gambar 4.5**

**Gambar Database Nomor Resi Pengiriman**

**Bentuk Gambar :**



**Gambar 4.6**

**Bentuk Gambar Menginput Nomor Resi Pengiriman**

**D. SP - 4 (Menambah dan Mengelola Data Produk)**

Pada Algoritma 4.4, sistem memulai proses dengan menghubungkan diri ke database (01) dan memeriksa keberadaan produk yang sudah ada menggunakan slug dari input (02). Jika produk dengan slug tersebut sudah ditemukan (03), sistem akan menghentikan operasi dan melemparkan kesalahan "Conflict" dengan pesan bahwa produk tersebut sudah ada (04 - 05) demi mencegah duplikasi data. Namun, jika slug belum terpakai, sistem melanjutkan dengan membuat produk baru menggunakan data yang diinputkan (07). Setelah produk terbentuk, sistem memeriksa nilai stok awal; jika jumlah stok lebih besar dari 0 (08), sistem akan membuat catatan pergerakan stok (stock movement) secara otomatis (09). Dalam catatan ini, sistem melengkapi detail produk seperti ID, nama, dan kode (10-12), serta menetapkan tipe pergerakan sebagai 'in' (13) dan jumlah stoknya (14). Sistem juga mencatat alasan "Produk baru ditambahkan" (16), tipe referensi 'initial' (17), serta identitas pengguna yang melakukan penambahan tersebut (19-20). Perubahan stok direkam dari posisi 0 (21) menjadi jumlah stok baru (22) disertai catatan tambahan (23). Akhirnya, sistem mengembalikan status sukses beserta objek produk yang telah berhasil dibuat (25) sebagai tanda selesainya proses.

**Algoritma 4.4 Contoh Algoritma Menambah Data Produk**

1. CONNECT to database

**Algoritma 4.4 (Lanjutan)**

1. GET existing product by slug from input
2. IF product already exists THEN
3. THROW CONFLICT error: 'Product with this slug
4. already exists'
5. END IF
6. CREATE new product with input data
7. IF input stock is greater than 0 THEN
8. CREATE stock movement record:
9. Set productId to new product's ID
10. Set productName to new product's name
11. Set productCode to new product's slug
12. Set movementType to 'in'
13. Set quantity to input stock
14. Set unit to new product's unit
15. Set reason to 'Produk baru ditambahkan'
16. Set referenceType to 'initial'
17. Set referenceId to new product's ID as string
18. Set performedBy to current user's ID
19. Set performedByName to current user's name
20. Set previousStock to 0
21. Set newStock to input stock
22. Set notes to 'Stok awal saat produk dibuat'
23. END IF
24. RETURN success: true and the created product object

Pada Algoritma 4.5, sistem memulai proses dengan menghubungkan diri ke database (01) serta mengekstrak data ID, alasan penyesuaian stok, dan data pembaruan dari input (02). Selanjutnya, sistem memeriksa apakah terdapat data slug dalam pembaruan (04); jika ada, sistem mencari produk lain yang memiliki slug yang sama namun dengan ID yang berbeda (05). Apabila produk duplikat tersebut ditemukan (07), sistem akan menghentikan operasi dan melemparkan pesan "Conflict" (08) untuk mencegah duplikasi data. Kemudian, sistem memeriksa apakah terdapat nilai stok dalam data pembaruan (13). Jika ya, sistem mengambil data produk saat ini (14) dan mencatat stok lama (18). Jika stok lama berbeda dengan stok baru (19), sistem menghitung selisihnya (20) serta menentukan jenis pergerakan ('in' jika bertambah, 'out' jika berkurang) (22). Berdasarkan perhitungan tersebut, sistem membuat catatan pergerakan stok (25) yang mencakup detail produk, jumlah selisih, alasan penyesuaian (32), referensi tindakan, serta identitas pengguna yang melakukan perubahan (36-37). Setelah penanganan stok selesai, sistem memperbarui produk di database menggunakan data baru (43). Jika produk tidak ditemukan saat pembaruan (45), sistem melemparkan error, namun jika berhasil, sistem mengembalikan status sukses beserta objek produk yang telah diperbarui (48).

**Algoritma 4.5 Contoh Algoritma Mengelola Data Produk**

1. CONNECT to database
2. EXTRACT id, stockAdjustmentReason, and updateData from
3. input
4. IF updateData contains a slug THEN
5. FIND existing product with the same slug but different
6. ID
7. IF existing product found THEN
8. THROW CONFLICT error: 'Product with this slug already
9. exists'
10. END IF
11. END IF
12. SET previousStock to 0
13. IF updateData contains a stock value THEN
14. FIND current product by ID
15. IF product not found THEN
16. THROW NOT\_FOUND error: 'Product not found'
17. END IF
18. SET previousStock to current product's stock
19. IF previousStock is not equal to updateData.stock THEN
20. CALCULATE stockDifference = updateData.stock –
21. previousStock
22. SET movementType to 'in' if stockDifference > 0,
23. otherwise 'out'
24. SET quantity to absolute value of stockDifference
25. CREATE stock movement record:
26. Set productId to current product's ID
27. Set productName to current product's name
28. Set productCode to current product's slug
29. Set movementType
30. Set quantity
31. Set unit to current product's unit
32. Set reason to stockAdjustmentReason or 'Penyesuaian
33. stok manual'
34. Set referenceType to 'adjustment'
35. Set referenceId to a unique adjustment ID
36. Set performedBy to current user's ID
37. Set performedByName to current user's name
38. Set previousStock
39. Set newStock to updateData.stock
40. Set notes to stockAdjustmentReason
41. END IF
42. END IF
43. UPDATE product by ID with updateData, return the new
44. document and run validators

**Algoritma 4.5 (Lanjutan)**

1. IF updated product not found THEN
2. THROW NOT\_FOUND error: 'Product not found'
3. END IF
4. RETURN success: true and the updated product object

Pada Segmen Program 4.4, sistem akan mencari product di database berdasarkan inputan slug (02 - 04). Jika product yang dicari ada didalam database maka sistem akan memberikan pesan error “Product with this slug already exists” (05 - 10). Jika product belum ada, maka sistem akan membuat product baru dengan data berdasarkan inputan yang diterima (11). Jika stok yang diinputkan lebih dari 0 maka sistem akan membuat stockMovement baru dengan isi data berdasarkan produk yang baru dibuat (12 - 31). Kemudian sistem akan mengembalikan object yang berisi success dan data product yang barusan dibuat (32 - 35).

**Segmen Program 4.4 Contoh Program Menambah Data Produk**

1. …
2. const existingProduct = await Product.findOne({
3. slug: input.slug
4. });
5. if (existingProduct) {
6. throw new TRPCError({
7. code: 'CONFLICT',
8. message: 'Product with this slug already exists',
9. });
10. }
11. const product = await Product.create(input);
12. if (input.stock > 0) {
13. await StockMovement.create({
14. productId: product.\_id as unknown as
15. import('mongoose').Types.ObjectId,
16. productName: product.name,
17. productCode: product.slug,
18. movementType: 'in',
19. quantity: input.stock,
20. unit: product.unit,
21. reason: 'Produk baru ditambahkan',
22. referenceType: 'initial',
23. referenceId: (product.\_id as unknown as
24. import('mongoose').Types.ObjectId).toString(),
25. performedBy: ctx.user.id,
26. performedByName: ctx.user.name,
27. previousStock: 0,
28. newStock: input.stock,

**Segmen Program 4.4 (Lanjutan)**

1. notes: 'Stok awal saat produk dibuat',
2. });
3. }
4. return {
5. success: true,
6. product: product.toObject(),
7. };
8. …

Pada Segmen Program 4.5, sistem akan mengubah stok product sebelumnya menjadi stok product saat ini (02). Jika stok product sebelumnya tidak cocok dengan stok updateData (03) maka sistem akan membuat stockDifference, movementType, dan quantity (04 - 08). Kemudian sistem akan membuat StockMovement baru yang datanya berisi data dari produk saat ini (09 - 26). Setelah itu sistem akan mencari product berdasarkan productid dan akan mengupdate data dari product tersebut (27 - 31). Jika product tidak ditemukan di database maka sistem akan memberikan pesan error “Product not found” (32 - 37). Terakhir sistem akan mengembalikan object yang berisi success dan data product (38 - 41).

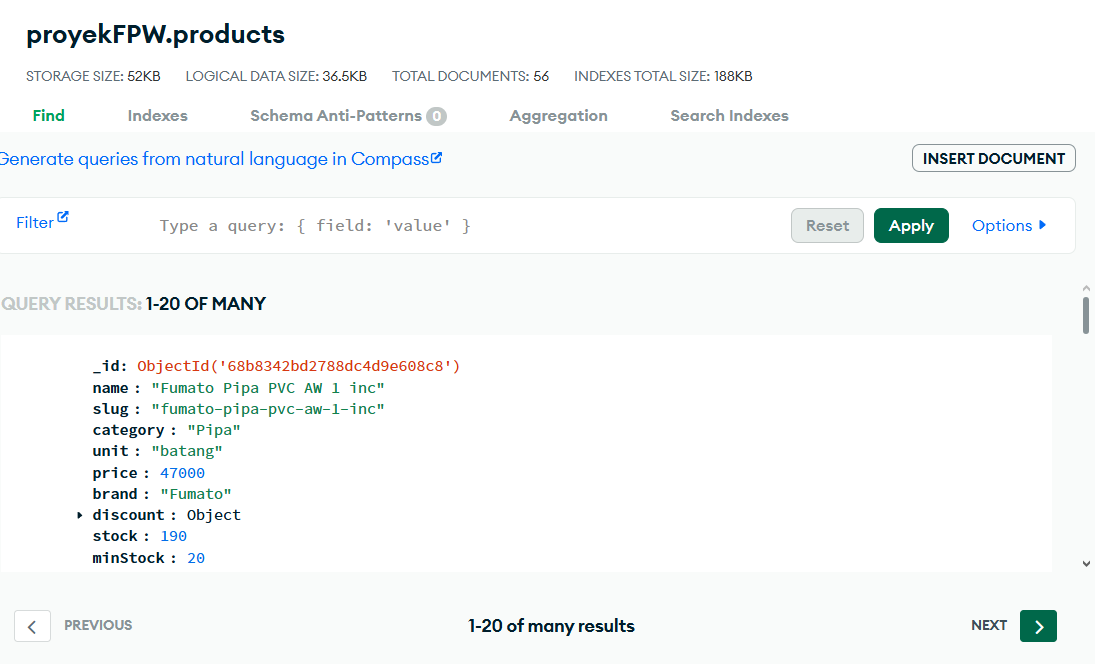
**Segmen Program 4.5 Contoh Program Mengelola Data Produk**

1. …
2. previousStock = currentProduct.stock;
3. if (previousStock !== updateData.stock) {
4. const stockDifference = updateData.stock –
5. previousStock;
6. const movementType = stockDifference > 0 ?
7. 'in' : 'out';
8. const quantity = Math.abs(stockDifference);
9. await StockMovement.create({
10. productId: currentProduct.\_id,
11. productName: currentProduct.name,
12. productCode: currentProduct.slug,
13. movementType,
14. quantity,
15. unit: currentProduct.unit,
16. reason: stockAdjustmentReason ||
17. 'Penyesuaian stok manual',
18. referenceType: 'adjustment',
19. referenceId: `ADJ-${Date.now()}`,
20. performedBy: ctx.user.id,
21. performedByName: ctx.user.name,
22. previousStock,

**Segmen Program 4.5 (Lanjutan)**

1. newStock: updateData.stock,
2. notes: stockAdjustmentReason,
3. });
4. }
5. const product = await Product.findByIdAndUpdate(
6. id,
7. { $set: updateData },
8. { new: true, runValidators: true }
9. );
10. if (!product) {
11. throw new TRPCError({
12. code: 'NOT\_FOUND',
13. message: 'Product not found',
14. });
15. }
16. return {
17. success: true,
18. product: product.toObject(),
19. };
20. …

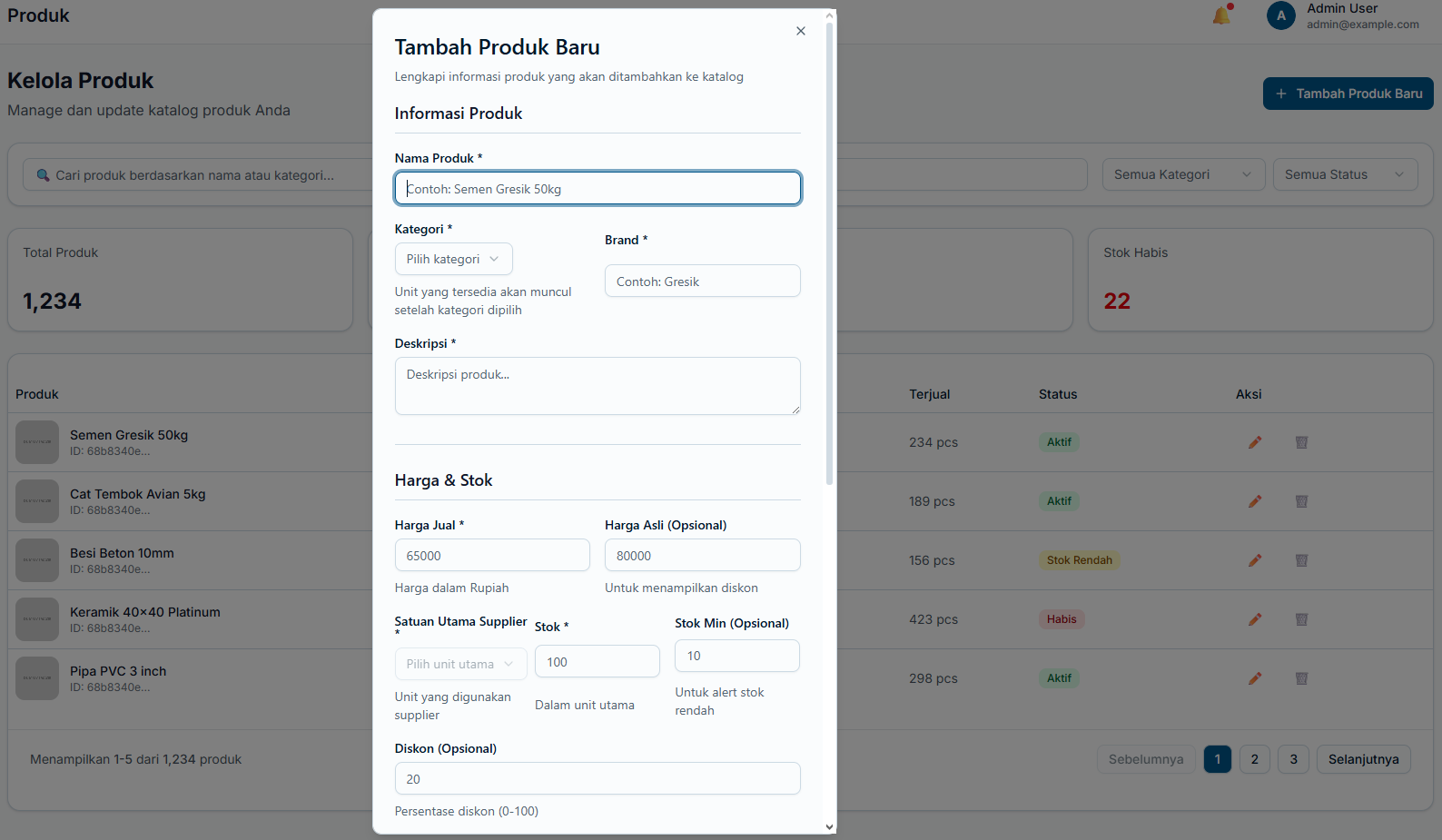
**Database yang Digunakan :**



**Gambar 4.7**

**Gambar Database Menambah dan Mengelola Data Produk**

**Bentuk Gambar :**



**Gambar 4.8**

**Bentuk Gambar Menambah dan Mengelola Data Produk**

**E. SP - 5 (Melihat Laporan Transaksi, Inventori, dan Data Pengguna)**

Pada Algoritma 4.6, sistem memulai proses dengan menghubungkan diri ke database (01) dan menginisialisasi waktu saat ini sebagai tanggal akhir (02-03) serta menyiapkan variabel label periode (04). Selanjutnya, sistem menentukan rentang waktu berdasarkan jenis periode yang dipilih (05). Jika memilih 'daily', sistem mengatur tanggal mulai ke 7 hari sebelumnya (07); jika 'weekly', mundur 8 minggu (10); dan jika 'monthly', mundur 12 bulan (13). Untuk pilihan 'custom' (15), sistem memvalidasi kelengkapan tanggal input (16), dan jika valid, menggunakan tanggal tersebut (20-21), namun jika tipe periode tidak dikenali, sistem akan melempar error (24). Setelah rentang waktu ditetapkan, sistem mengambil seluruh pesanan yang berstatus 'paid' dalam periode tersebut (26), lalu menghitung total pendapatan (28), jumlah total pesanan (29), dan total produk terjual (30). Sistem kemudian menyiapkan array kosong untuk data grafik (32) dan mengisinya berdasarkan logika periode: melakukan perulangan harian untuk 'daily' (33), mingguan untuk 'weekly' (36), atau bulanan untuk 'monthly' (39). Khusus untuk periode 'custom', sistem menghitung selisih hari (42); jika rentangnya 31 hari atau kurang, data disusun secara harian (44), namun jika lebih lama, data dikelompokkan per bulan (48). Akhirnya, sistem mengembalikan objek lengkap berisi statistik ringkasan dan data visualisasi grafik (53 - 54).

**Algoritma 4.6 Contoh Algoritma Lihat Laporan Transaksi**

1. CONNECT to database
2. SET now to current date and time
3. SET endDate to now
4. SET periodLabel to empty strin
5. SWITCH input.period:
6. CASE 'daily':
7. SET startDate to 7 days before now
8. SET periodLabel to '7 Hari Terakhir'
9. CASE 'weekly':
10. SET startDate to 8 weeks before now
11. SET periodLabel to '8 Minggu Terakhir'
12. CASE 'monthly':
13. SET startDate to 12 months before now
14. SET periodLabel to '12 Bulan Terakhir'
15. CASE 'custom':
16. IF input.startDate or input.endDate is missing THEN
17. THROW BAD\_REQUEST error: 'Custom period memerlukan
18. startDate dan endDate'
19. END IF
20. SET startDate to input.startDate
21. SET endDate to input.endDate
22. SET periodLabel to formatted date range
23. DEFAULT:
24. THROW BAD\_REQUEST error: 'Invalid period type'
25. END SWITCH
26. GET orders from database within startDate and endDate with
27. paymentStatus 'paid'
28. CALCULATE totalRevenue by summing order totals
29. CALCULATE totalOrders as count of orders
30. CALCULATE totalProductsSold by summing item quantities across
31. all orders
32. CREATE empty chartData array
33. IF input.period is 'daily' THEN
34. LOOP for last 7 days:
35. CALCULATE daily orders and add to chartData
36. ELSE IF input.period is 'weekly' THEN
37. LOOP for last 8 weeks:
38. CALCULATE weekly orders and add to chartData
39. ELSE IF input.period is 'monthly' THEN
40. LOOP for last 12 months:
41. CALCULATE monthly orders and add to chartData
42. ELSE (custom period):

**Algoritma 4.6 (Lanjutan)**

1. CALCULATE daysDifference between startDate and endDate
2. IF daysDifference <= 31 THEN
3. LOOP for each day in range:
4. CALCULATE daily orders and add to chartData
5. ELSE (daysDifference > 31) THEN
6. GROUP orders by month:
7. CALCULATE monthly revenue and orders
8. ADD to chartData, sorted by date
9. END IF
10. END IF
11. RETURN object containing totalRevenue, totalOrders,
12. totalProductsSold, chartData, periodLabel, and dateRange

Pada Segmen Program 4.6, sistem akan menyimpan waktu saat ini ke dalam variabel (02), dan menyimpan startDate, endDate, dan periodLabel (03 - 05). Kemudian sistem akan mencari order dari database yang difilter dalam kurun waktu createdAt dan statusnya sudah ‘paid’ yang nantinya akan dilean (06 - 09). Sistem juga akan menentukan totalRevenue berdasarkan order, totalOrder berdasarkan banyaknya order, totalProductsSold berdasarkan order, dan juga array chartData (10 - 25). Kemudian langkah terakhir, sistem akan mengembalikan object yang berisi totalRevenue, totalOrders, totalProductsSold, chartData, periodLabel, dan juga dateRange (27 - 36).

**Segmen Program 4.6 Contoh Program Lihat Laporan Transaksi**

1. …
2. const now = new Date();
3. let startDate: Date;
4. let endDate = new Date();
5. let periodLabel = '';
6. const orders = await Order.find({
7. createdAt: { $gte: startDate, $lte: endDate },
8. paymentStatus: 'paid',
9. }).lean();
10. const totalRevenue = orders.reduce((sum, order) => sum +
11. order.total, 0);
12. const totalOrders = orders.length;
13. const totalProductsSold = orders.reduce(
14. (sum, order) =>
15. sum + order.items.reduce((itemSum: number, item: {
16. quantity: number }) =>
17. itemSum + item.quantity, 0),
18. 0

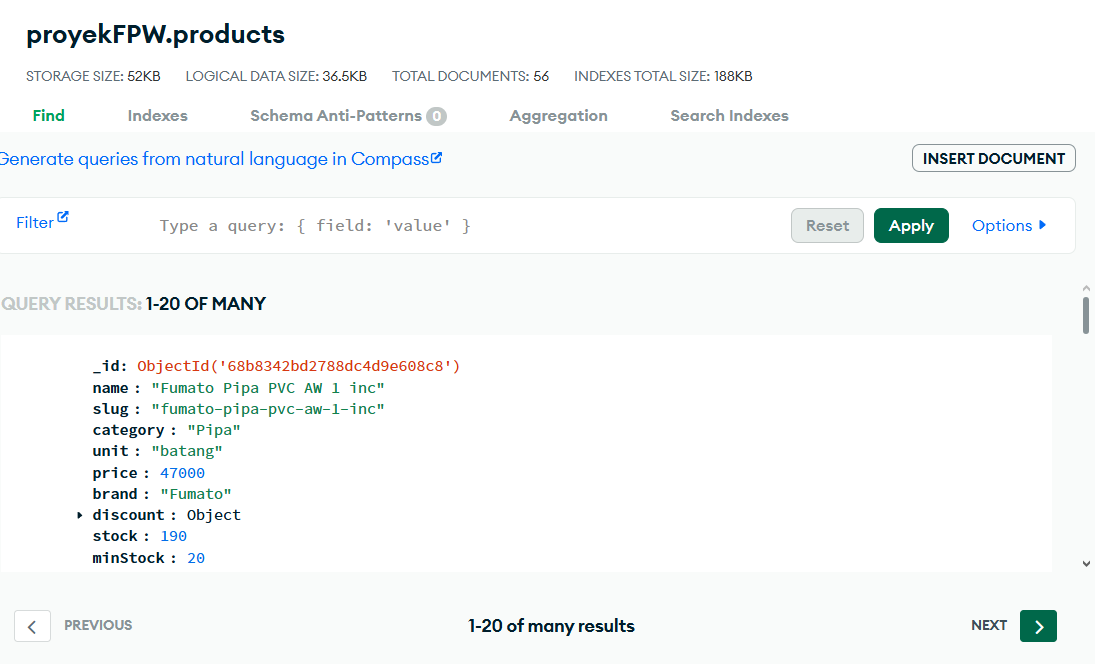
**Segmen Program 4.6 (Lanjutan)**

1. );
2. const chartData: Array<{
3. period: string;
4. revenue: number;
5. orders: number;
6. date: string;
7. }> = [];
8. …
9. return {
10. totalRevenue,
11. totalOrders,
12. totalProductsSold,
13. chartData,
14. periodLabel,
15. dateRange: {
16. start: startDate.toISOString(),
17. end: endDate.toISOString(),
18. }

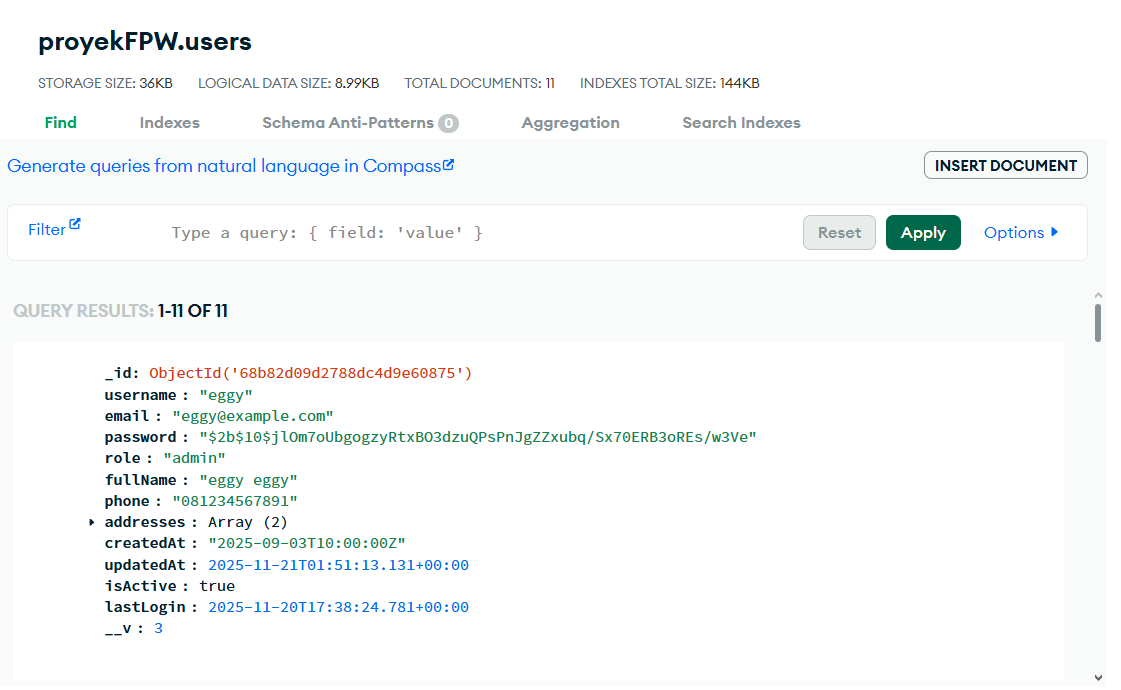
**Database yang Digunnakan :**



**(a)**



**(b)**

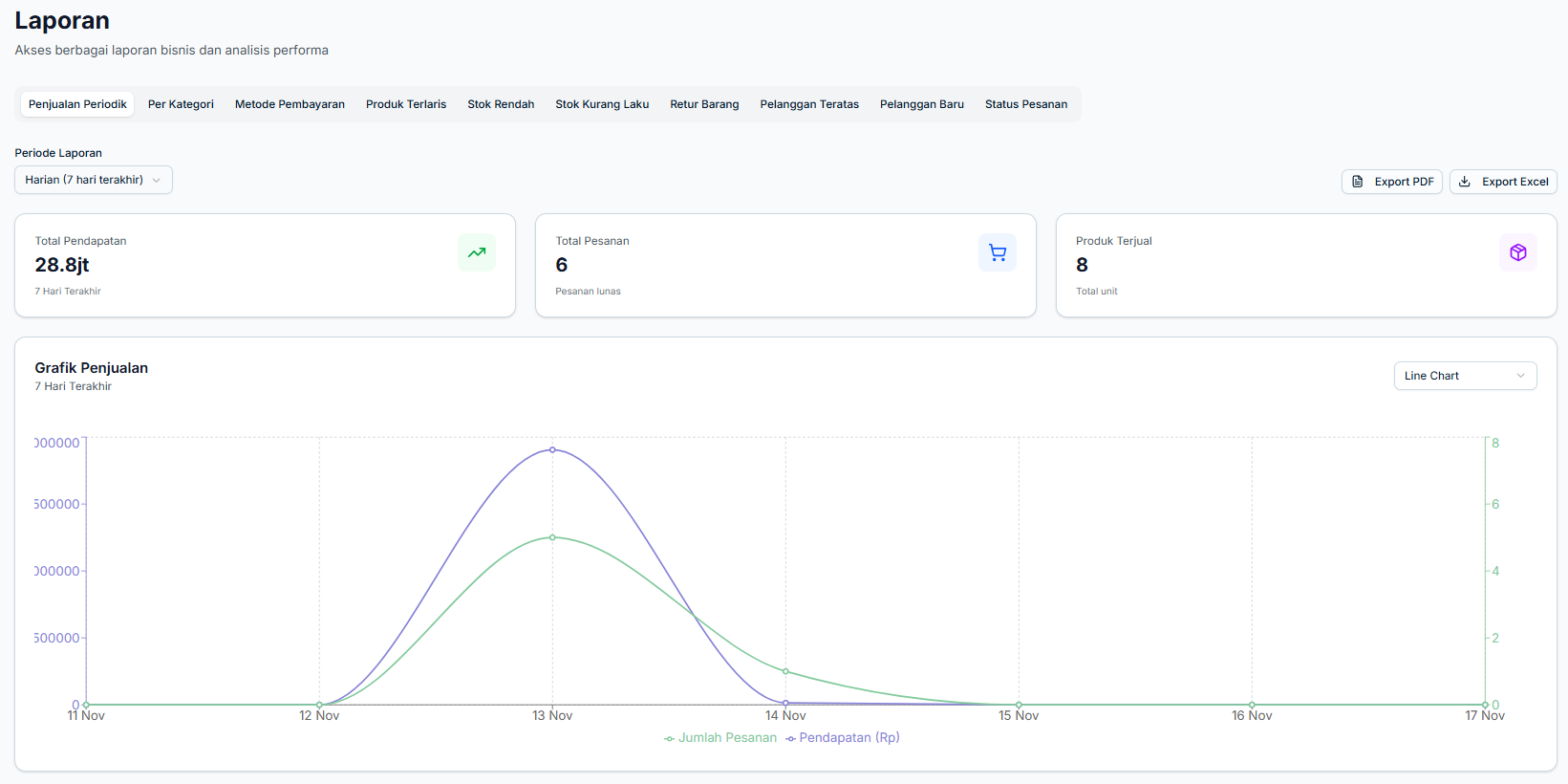


**(c)**

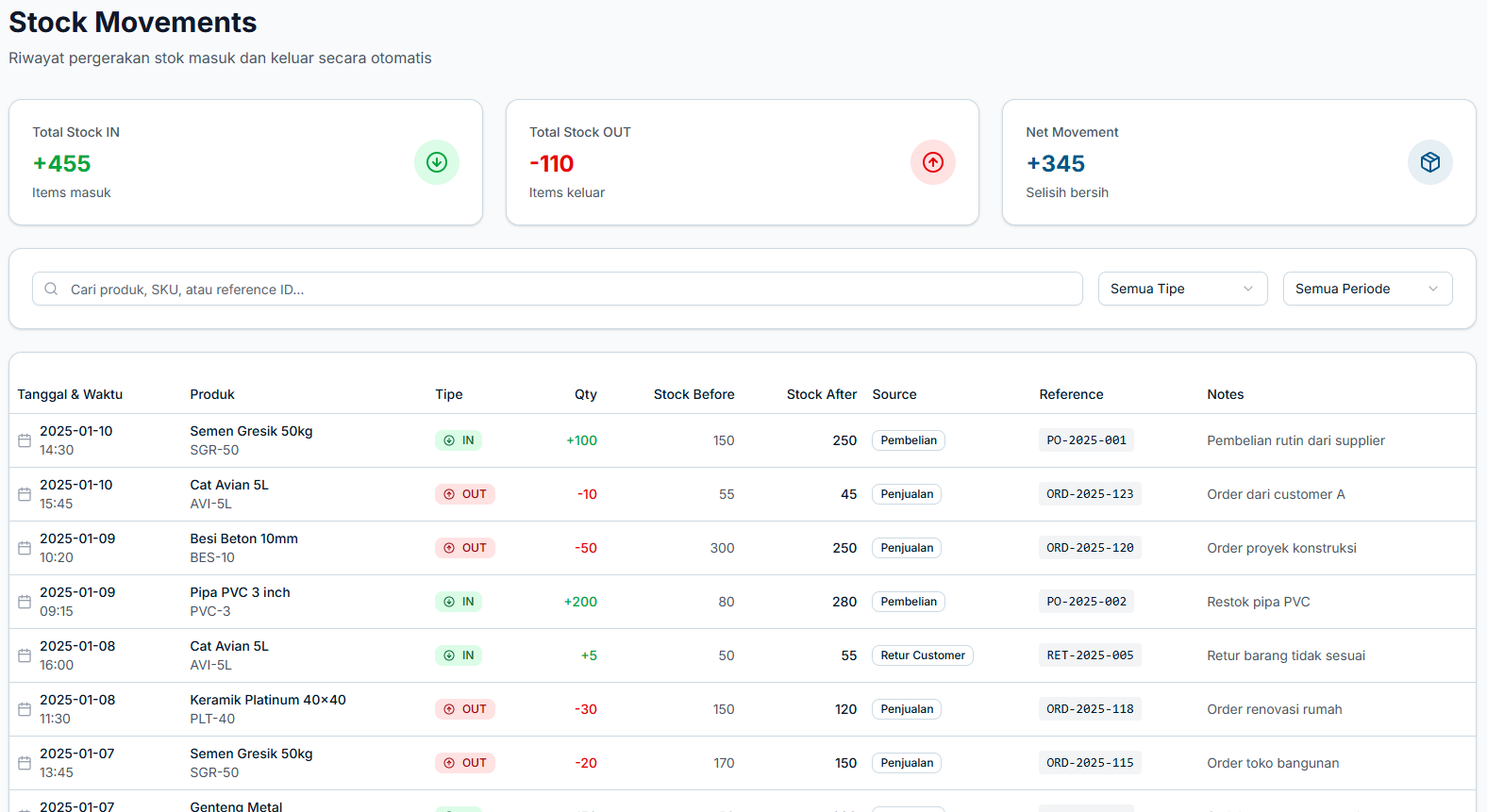
**Gambar 4.9**

**Gambar Database (a) Transaksi. (b) Inventori. (c) Data Pengguna**

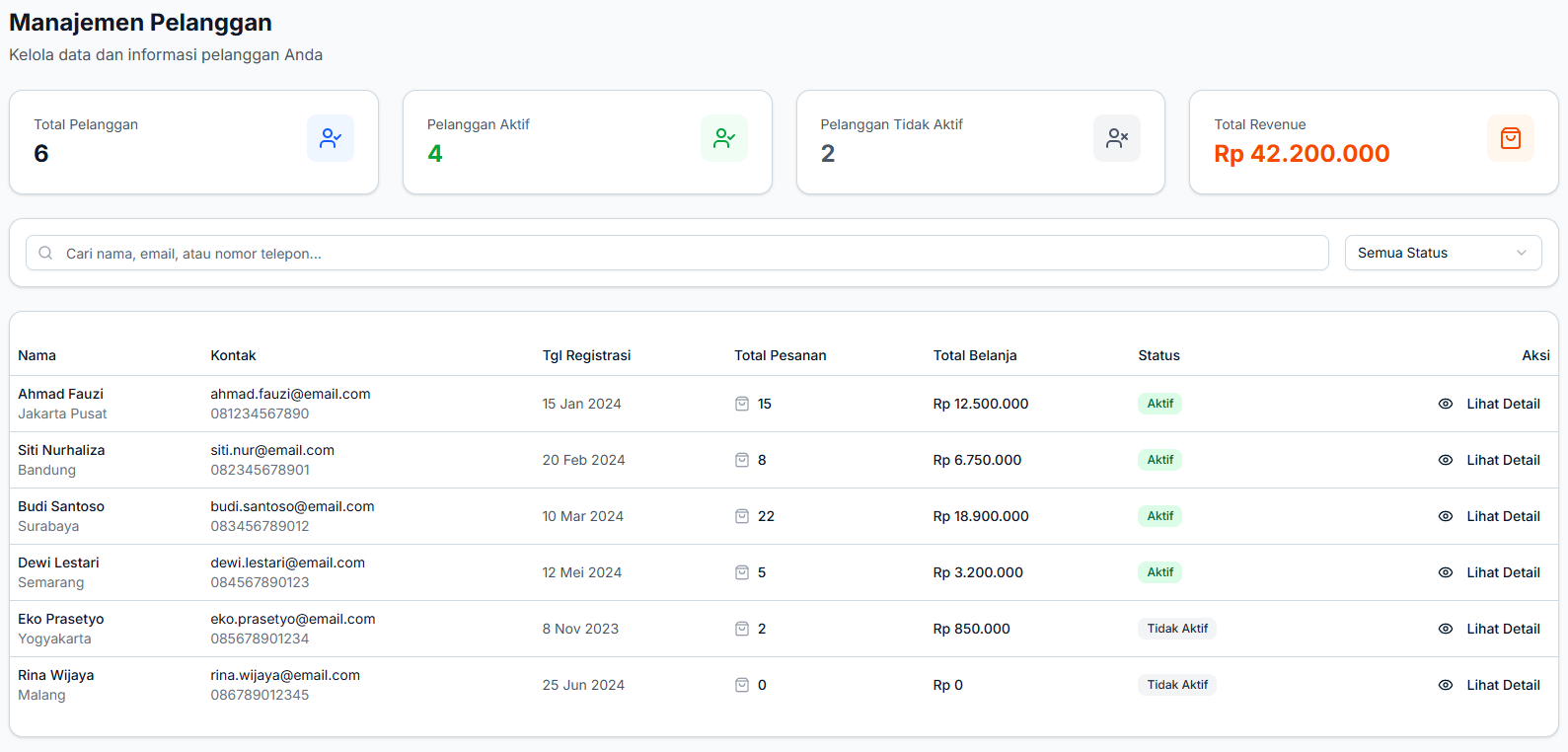
**Bentuk Gambar :**



**(a)**



**(b)**



**(c)**

**Gambar 4.10**

**Bentuk Gambar Laporan (a) Transaksi. (b) Inventori. (c) Data Pengguna**

**E. SP - 6 (Menambah dan Mengelola Data Voucher)**

Pada Algoritma 4.7, sistem akan memulai prosedur pembuatan voucher (01 - 02) dengan memasuki blok eksekusi aman (try-catch) (03). Langkah pertama adalah verifikasi keamanan, di mana sistem memeriksa apakah pengguna yang memanggil fungsi ini memiliki peran sebagai 'admin' (04). Jika bukan admin, akses segera ditolak dengan pesan kesalahan Forbidden (05). Setelah koneksi database dipastikan aktif (07), sistem melakukan validasi integritas data dengan mencari apakah kode voucher yang diinputkan sudah pernah ada sebelumnya (08 - 09). Jika kode duplikat ditemukan, sistem menghentikan proses untuk mencegah konflik data (10 - 13). Sistem kemudian melakukan validasi logika bisnis dengan memastikan bahwa tanggal berakhir voucher tidak boleh lebih awal dari tanggal mulai (14 - 17). Jika semua validasi lolos, sistem menginisialisasi objek voucher baru dengan menggabungkan data input admin dan nilai default sistem, seperti usedCount diset ke 0 dan isActive diset ke true (18 - 20). Objek baru ini kemudian disimpan secara permanen ke dalam koleksi Voucher di database (21). Jika penyimpanan berhasil, sistem mengembalikan status sukses beserta data voucher yang baru dibuat (22). Apabila terjadi kesalahan teknis selama proses, sistem akan menangkap error tersebut, mencatatnya dalam log (24), dan mengirimkan respon kegagalan kepada pengguna (25).

**Algoritma 4.7 Contoh Algoritma Menambah Data Voucher**

1. BEGIN createVoucher (input: code, name, type, value, dates,
2. limits)
3. TRY
4. IF ctx.user.role IS NOT 'admin' THEN
5. THROW TRPCError (Code: FORBIDDEN, Message: "Only admin...")
6. END IF
7. CALL connectDB()
8. GET existingVoucher FROM Database Voucher WHERE code =
9. input.code
10. IF existingVoucher EXISTS THEN
11. THROW TRPCError (Code: CONFLICT, Message: "Voucher code
12. exists")
13. END IF
14. IF input.endDate <= input.startDate THEN
15. THROW TRPCError (Code: BAD\_REQUEST, Message: "Invalid date
16. range")
17. END IF
18. SET newVoucher = NEW Voucher({
19. ...input, usedCount: 0, isActive: TRUE, createdAt: NOW
20. })
21. SAVE newVoucher TO Database
22. RETURN { success: TRUE, voucher: newVoucher }
23. CATCH error
24. LOG error to Console
25. THROW TRPCError (Code: INTERNAL\_SERVER\_ERROR)
26. END

Pada Algoritma 4.8, sistem akan memulai prosedur pembaruan data voucher (01 - 02) dengan memasuki blok eksekusi aman (03) dan melakukan verifikasi hak akses pengguna. Sistem memastikan bahwa hanya pengguna dengan peran 'admin' yang diizinkan melanjutkan proses; jika tidak, akses ditolak dengan status Forbidden (04 - 06). Setelah terhubung ke database (07), sistem mencari keberadaan voucher yang akan diedit berdasarkan ID yang diberikan. Jika voucher tidak ditemukan, sistem menghentikan proses dengan pesan Not Found (08 - 12). Sistem kemudian melakukan validasi integritas data yang kompleks. Jika admin mengubah kode voucher, sistem memastikan kode baru tersebut belum digunakan oleh voucher lain untuk menghindari duplikasi data (12 - 16). Selain itu, sistem juga memvalidasi rentang tanggal (jika ada perubahan tanggal) untuk memastikan tanggal berakhir tidak mendahului tanggal mulai (19 - 22). Setelah semua validasi terpenuhi, sistem melakukan operasi pembaruan data pada dokumen voucher terkait dan memperbarui stempel waktu updatedAt (23 - 24). Hasil pembaruan kemudian dikembalikan sebagai respon sukses (25). Jika terjadi kendala teknis selama proses, sistem menangkap kesalahan tersebut, mencatatnya dalam log konsol (27), dan mengirimkan notifikasi kegagalan kepada pengguna (28).

**Algoritma 4.8 Contoh Algoritma Mengelola Data Voucher**

1. BEGIN updateVoucher (input: id, code, name, type, value, dates,
2. etc.)
3. TRY
4. IF ctx.user.role IS NOT 'admin' THEN
5. THROW TRPCError (Code: FORBIDDEN, Message: "Only admin...")
6. END IF
7. CALL connectDB()
8. GET voucher FROM Database Voucher WHERE \_id = input.id
9. IF voucher NOT EXISTS THEN
10. THROW TRPCError (Code: NOT\_FOUND, Message: "Voucher not
11. found")
12. END IF
13. IF input.code IS SET AND input.code != voucher.code THEN
14. GET existing FROM Database WHERE code = input.code
15. IF existing EXISTS THEN
16. THROW TRPCError (Code: CONFLICT, Message: "Code exists")
17. END IF
18. END IF
19. IF input.endDate <= input.startDate THEN
20. THROW TRPCError (Code: BAD\_REQUEST, Message: "Invalid date
21. range")
22. END IF
23. UPDATE Voucher SET ...input, updatedAt: NOW WHERE \_id =
24. input.id
25. RETURN { success: TRUE, voucher: updatedVoucher }
26. CATCH error
27. LOG error to Console
28. THROW TRPCError (Code: INTERNAL\_SERVER\_ERROR)
29. END

Pada Segmen Program 4.7, sistem akan mencari voucher di database berdasarkan inputan kode voucher yang diberikan (02), jika voucher tersebut terdapat dalam database maka sistem tidak akan membuat voucher tersebut dan memberikan pesan error “Voucher code already exists” (04 - 09). Jika code voucher belum ada di database maka, sistem akan menyimpan waktu saat ini kedalam sebuah variabel (11), kemudian sistem akan membuat voucher baru berdasarkan inputan yang diberikan beserta waktu yang telah disimpan (12 - 18). Kemudian sistem akan menyimpan voucher yang baru saja dibuat kedalam database (19) dan mengembalikan object success dan isi data dari voucher yang sudah dibuat (20).

**Segmen Program 4.7 Contoh Program Menambah Data Voucher**

1. …
2. const existingVoucher = await Voucher.findOne({ code: input.code
3. });
4. if (existingVoucher) {
5. throw new TRPCError({
6. code: 'CONFLICT',
7. message: 'Voucher code already exists',
8. });
9. }
10. …
11. const now = new Date().toISOString();
12. const voucher = new Voucher({
13. ...input,
14. usedCount: 0,
15. isActive: true,
16. createdAt: now,
17. updatedAt: now,
18. });
19. await voucher.save();
20. return { success: true, voucher };

Pada Segmen Program 4.8, sistem akan mencari voucher dari database berdasarkan id voucher (01), jika id voucher tersebut tidak ditemukan maka sistem akan memberikan pesan error “Voucher not found” (03 - 08). Kemudian sistem akan mengecek apakah masa penggunaan voucher apakah valid atau tidak, jika tidak valid maka sistem akan memberikan pesan error “End date must be after start date” (09 - 18). Kemudian sistem akan mencari sekaligus mengupdate voucher berdasarkan id voucher yang diberikan (19 – 26), dan langkah terakhir sistem akan mengembalikan object yang berisi success dan data voucher (27).

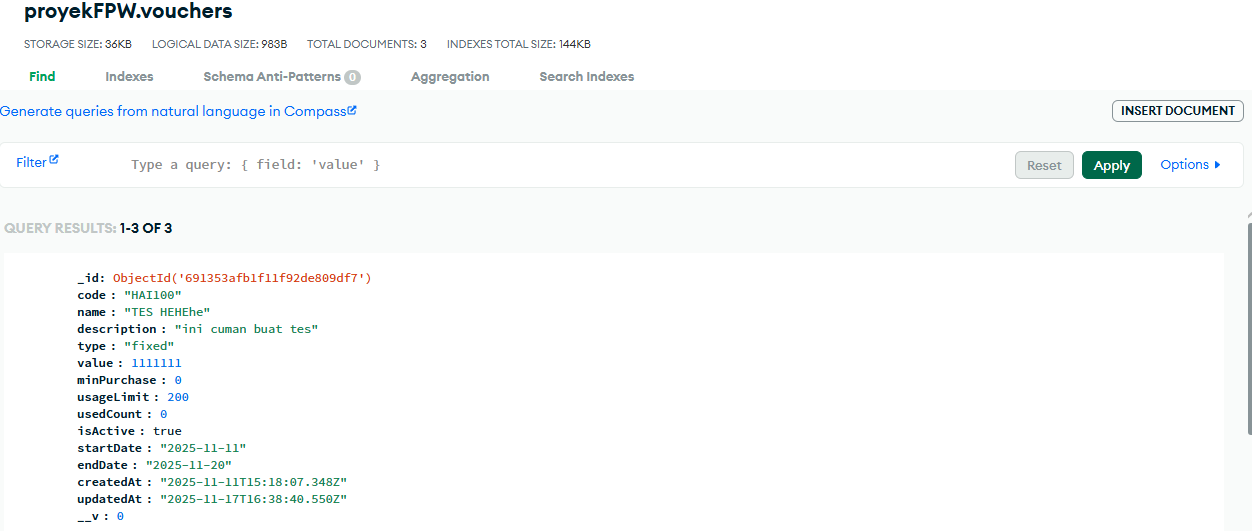
**Segmen Program 4.8 Contoh Program Mengelola Data Voucher**

1. …

**Segmen Program 4.8 (Lanjutan)**

1. const voucher = await Voucher.findById(id);
2. if (!voucher) {
3. throw new TRPCError({
4. code: 'NOT\_FOUND',
5. message: 'Voucher not found',
6. });
7. }
8. if (updateData.startDate || updateData.endDate) {
9. const startDate = updateData.startDate || voucher.startDate;
10. const endDate = updateData.endDate || voucher.endDate;
11. if (new Date(endDate) <= new Date(startDate)) {
12. throw new TRPCError({
13. code: 'BAD\_REQUEST',
14. message: 'End date must be after start date',
15. });
16. }
17. }
18. const updatedVoucher = await Voucher.findByIdAndUpdate(
19. id,
20. {
21. ...updateData,
22. updatedAt: new Date().toISOString(),
23. },
24. { new: true }
25. );
26. return { success: true, voucher: updatedVoucher };

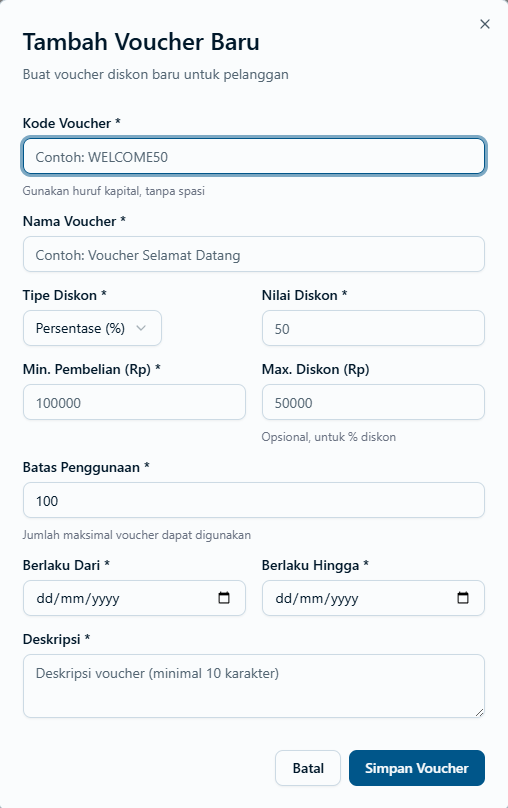
**Database yang Digunakan :**

****

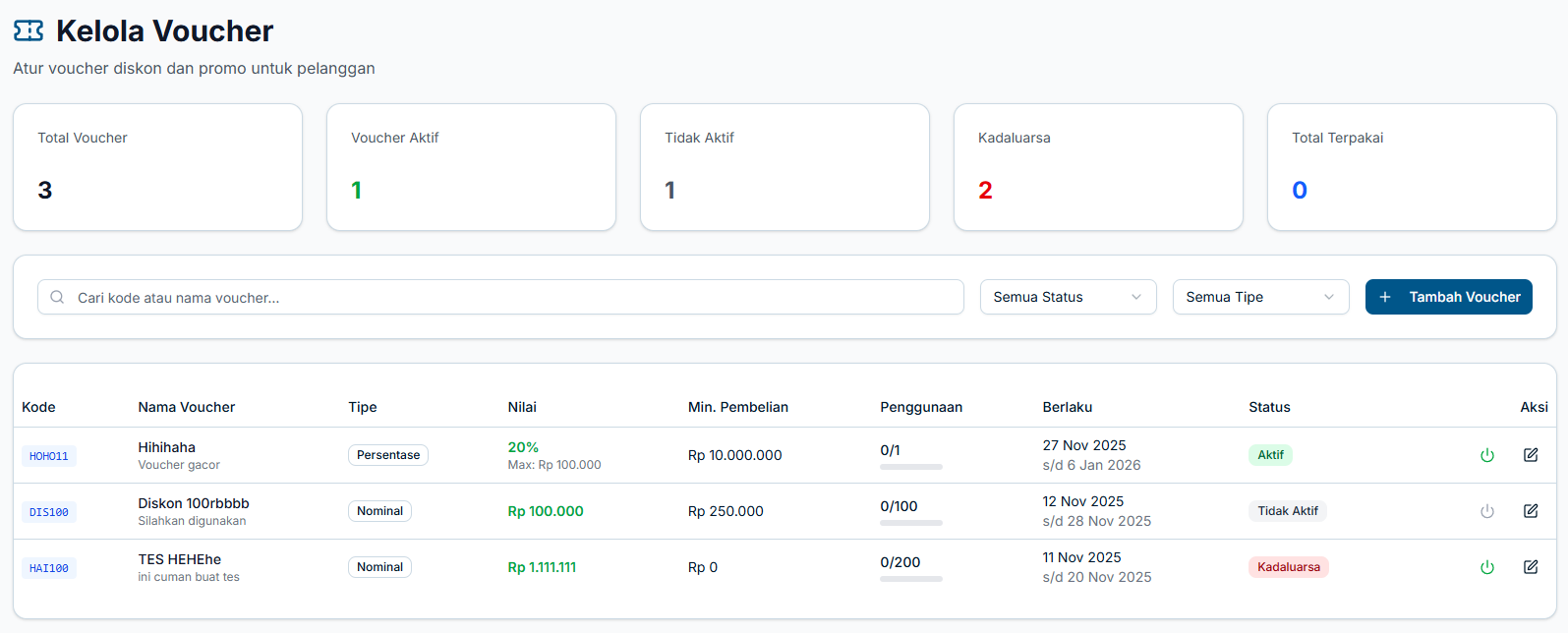
**Gambar 4.11**

**Gambar Database Voucher**

**Bentuk Gambar :**

****

**(a)**

****

**(b)**

**Gambar 4.12**

**Bentuk Gambar (a) Menambah Voucher. (b) Mengelola Voucher**

**E. SP - 7 (Menambah, Mengelola, dan Mengaktifasikan Akun Staff dan Admin)**

Pada Algoritma 4.9, sistem memulai prosedur pembuatan akun pegawai baru (01) dengan memasuki blok eksekusi aman (02). Langkah awal yang krusial adalah validasi otorisasi, di mana sistem memastikan bahwa pengguna yang melakukan permintaan ini memiliki peran sebagai 'admin'. Jika bukan admin, akses segera ditolak dengan status Forbidden (03 - 06). Selanjutnya, sistem melakukan validasi integritas data dengan memeriksa ketersediaan email di database. Jika email yang didaftarkan sudah digunakan oleh pengguna lain, sistem menghentikan proses dan mengembalikan pesan kesalahan konflik (07 - 11). Sebagai langkah keamanan data sensitif, sistem melakukan enkripsi (hashing) pada kata sandi yang diinputkan menggunakan algoritma bcrypt (12). Setelah itu, sistem menyusun objek data pegawai baru dengan menyertakan data input, kata sandi yang telah dienkripsi, serta atribut default seperti username (diambil dari bagian depan email) dan status isActive yang bernilai true (13 - 17). Data tersebut kemudian disimpan ke dalam koleksi Pengguna di database (18). Jika penyimpanan berhasil, sistem mengembalikan konfirmasi sukses beserta ID pegawai baru (19). Apabila terjadi kesalahan sistem selama proses berlangsung, sistem akan menangkap kesalahan tersebut, mencatatnya dalam log, dan memberikan notifikasi kegagalan kepada pengguna (21 - 22).

**Algoritma 4.9 Contoh Algoritma Menambah Akun Staff dan Admin**

1. BEGIN createStaff (input: fullName, email, phone, password, role)
2. TRY
3. IF ctx.user.role IS NOT 'admin' THEN
4. THROW TRPCError (Code: FORBIDDEN, Message: "Hanya
5. admin...")
6. END IF
7. GET existingUser FROM Database User WHERE email = input.email
8. IF existingUser EXISTS THEN
9. THROW TRPCError (Code: CONFLICT, Message: "Email sudah
10. terdaftar")
11. END IF
12. SET hashedPassword = CALL bcryptjs.hash(input.password, 10)
13. SET newStaff = NEW User({
14. ...input, password: hashedPassword,
15. username: input.email.split('@')[0],
16. isActive: TRUE, createdAt: NOW
17. })
18. SAVE newStaff TO Database
19. RETURN { success: TRUE, staffId: newStaff.\_id }
20. CATCH error
21. LOG error to Console
22. THROW TRPCError (Code: INTERNAL\_SERVER\_ERROR)

Pada Algoritma 4.10, Sistem memulai prosedur pengambilan daftar pegawai (01) dengan memasuki blok eksekusi aman (02). Langkah pertama adalah verifikasi otorisasi ketat, di mana sistem memastikan bahwa pengguna yang meminta data ini memiliki peran sebagai 'admin'. Jika pengguna bukan admin, sistem langsung menolak akses dengan status Forbidden (03 - 06) untuk mencegah kebocoran data internal kepada pihak yang tidak berwenang. Setelah lolos verifikasi, sistem melakukan query ke koleksi Pengguna di database untuk mengambil semua akun yang memiliki peran 'admin' atau 'staff' (07 - 08). Demi keamanan data, sistem secara eksplisit mengecualikan field kata sandi (password) dari hasil pencarian agar informasi sensitif tidak terekspos ke frontend (09). Data yang diambil kemudian diurutkan berdasarkan waktu pembuatan dari yang terbaru ke terlama (10). Daftar pegawai yang telah disaring dan diamankan ini kemudian dikembalikan kepada pemanggil fungsi (11). Jika terjadi kesalahan teknis selama proses pengambilan data, sistem akan menangkap kesalahan tersebut, mencatatnya dalam log, dan memberikan pesan kesalahan umum kepada pengguna (13-14).

**Algoritma 4.10 Contoh Algoritma Mengelola Akun Staff dan Admin**

1. BEGIN getAllStaff
2. TRY
3. IF ctx.user.role IS NOT 'admin' THEN
4. THROW TRPCError (Code: FORBIDDEN, Message: "Hanya
5. admin...")
6. END IF
7. GET staffList FROM Database User
8. WHERE role IN ['admin', 'staff']
9. EXCLUDE field password
10. SORT BY createdAt DESCENDING
11. RETURN staffList
12. CATCH error
13. LOG error to Console
14. THROW TRPCError (Code: INTERNAL\_SERVER\_ERROR)

Pada Algoritma 4.11, sistem memulai prosedur pengubahan status keaktifan akun (01) dengan memasuki blok eksekusi aman (02). Tahap pertama adalah verifikasi hak akses, di mana sistem memastikan pemanggil fungsi adalah seorang 'admin' (03 - 06). Selanjutnya, sistem melakukan proteksi diri dengan mencegah admin menonaktifkan akunnya sendiri, yang dapat menyebabkan admin terkunci dari sistem (07 - 10). Sistem kemudian mencari data pengguna target di database berdasarkan ID yang diberikan (11). Jika pengguna tidak ditemukan, proses dihentikan dengan pesan Not Found (12 - 15). Sebelum melakukan perubahan, sistem memvalidasi bahwa target pengguna benar-benar memiliki peran 'admin' atau 'staff'. Hal ini mencegah kesalahan sistem yang tidak sengaja mengubah status akun pelanggan (customer) melalui fitur ini (16 - 19). Jika validasi lolos, sistem melakukan operasi toggle (pembalikan nilai) pada status isActive—jika sebelumnya aktif menjadi non-aktif, dan sebaliknya—serta memperbarui waktu perubahan terakhir (20 - 21). Perubahan tersebut disimpan ke database (22) dan sistem mengembalikan status terbaru sebagai konfirmasi (23). Jika terjadi kendala teknis, sistem menangkap kesalahan, mencatat log, dan mengirimkan notifikasi kegagalan (25 - 26).

**Algoritma 4.11 Contoh Algoritma Mengaktifasikan Akun Staff dan Admin**

1. BEGIN toggleStaffStatus (input: userId)
2. TRY
3. IF ctx.user.role IS NOT 'admin' THEN
4. THROW TRPCError (Code: FORBIDDEN, Message: "Hanya
5. admin...")
6. END IF
7. IF input.userId == ctx.user.id THEN
8. THROW TRPCError (Code: FORBIDDEN, Message: "Tidak dapat
9. nonaktifkan diri sendiri")
10. END IF
11. GET staff FROM Database User WHERE \_id = input.userId
12. IF staff NOT EXISTS THEN
13. THROW TRPCError (Code: NOT\_FOUND, Message: "Staff tidak
14. ditemukan")
15. END IF
16. IF staff.role NOT IN ['admin', 'staff'] THEN
17. THROW TRPCError (Code: BAD\_REQUEST, Message: "Hanya status
18. admin/staff...")
19. END IF
20. SET staff.isActive = NOT staff.isActive
21. SET staff.updatedAt = NOW
22. SAVE staff TO Database
23. RETURN { success: TRUE, isActive: staff.isActive }
24. CATCH error
25. LOG error to Console
26. THROW TRPCError (Code: INTERNAL\_SERVER\_ERROR)

Pada Segmen Program 4.9, sistem akan mengecek apakah user yang ingin menambahkan akun staff baru adalah admin atau tidak (02), jika bukan admin maka sistem akan mengirimkan error “FORBIDDEN” (03 - 07). Kemudian sistem akan mengecek apakah email yang diberikan sudah ada didalam database atau belum, jika sudah ada maka sistem akan memberikan pesan error “Email sudah terdaftar” (08 - 14). Ketika semua proses pengecekan berjalan dengan lancer, maka inputan password user baru akan dihash demi menjaga keamanan akun (15) dan setelah dihash, sistem akan membuat akun staff baru berdasarkan inputan user (16 - 27). Langkah terakhir sistem akan mengembalikan status success dan pesan berhasil, serta id staff yang baru saja dibuat (28 - 32).

**Segmen Program 4.9 Contoh Program Menambah Akun Staff dan Admin**

1. if (ctx.user.role !== 'admin') {
2. throw new TRPCError({
3. code: 'FORBIDDEN',
4. message: 'Hanya admin yang dapat menambahkan staff',
5. });
6. }
7. const existingUser = await User.findOne({ email: input.email });
8. if (existingUser) {
9. throw new TRPCError({
10. code: 'CONFLICT',
11. message: 'Email sudah terdaftar',
12. });
13. }
14. const hashedPassword = await bcryptjs.hash(input.password, 10);
15. const newStaff = await User.create({
16. fullName: input.fullName,
17. email: input.email,
18. username: input.email.split('@')[0],
19. phone: input.phone,
20. password: hashedPassword,
21. role: input.role,
22. addresses: [],
23. isActive: true,
24. createdAt: new Date(),
25. updatedAt: new Date(),
26. });
27. return {
28. success: true,
29. message: 'Staff berhasil ditambahkan',
30. staffId: newStaff.\_id,
31. };

Pada Segmen Program 4.10, sistem akan pertama - tama mengecek apakah user yang berusaha mengakses apakah seorang admin atau tidak, jika bukan admin maka sistem akan memberikan pesan error “Hanya admin yang dapat meliaht daftar staff” (02 - 07). Ketika sudah melewati proses pengecekan role, maka sistem akan mencari user - user yang memiliki role “Admin” dan “Staff” dari database, tetapi tetap menjaga privasi tiap user dengan mengeluarkan password, serta mengurutkannya berdasarkan waktu pembuatan secara descending, dan akan dilean (08 - 13). Terakhir sistem akan mengembalikan staffList yang sudah dicari dari database (14).

**Segmen Program 4.10 Contoh Program Mengelola Akun Staff dan Admin**

1. …
2. if (ctx.user.role !== ‘admin’) {
3. throw new TRPCError({
4. code: ‘FORBIDDEN’,
5. Message: ‘Hanya admin yang dapat melihat daftar staff’,
6. });
7. }
8. const staffList = await User.find({
9. role: { $in: [‘admin’, ‘staff’] },
10. })
11. .select(‘-password’)
12. .sort({ createdAt: -1 })
13. .lean();
14. return staffList;

Pada Segmen Program 4.11, sistem akan memastikan bahwa user saat ini tidak bisa menonaktifkan akunnya sendiri (01 - 07). Kemudian sistem akan mencari akun staff dari database berdasarkan inputan userId yang diberikan (08), dan jika akun tersebut tidak ditemukan di database, maka sistem akan memberikan pesan error “Staff tidak ditemukan” (09 - 14). Langkah selanjutnya, sistem akan menonaktifkan akun staff yang sudah ditemukan (16) dan mengubah waktu updateAtnya menjadi waktu saat ini (17), dan selanjutnya akan disimpan oleh sistem (18). Langkah terakhir sistem akan mengembalikan status success dan juga pesan akun staff yang telah berubah (19 - 24).

**Segmen Program 4.11 Contoh Program Mengaktifasikan Akun Staff dan Admin**

1. if (input.userId === ctx.user.id) {
2. throw new TRPCError({

**Segmen Program 4.11 (Lanjutan)**

1. code: 'FORBIDDEN',
2. message: 'Anda tidak dapat menonaktifkan akun Anda
3. sendiri',
4. });
5. }
6. const staff = await User.findById(input.userId);
7. if (!staff) {
8. throw new TRPCError({
9. code: 'NOT\_FOUND',
10. message: 'Staff tidak ditemukan',
11. });
12. }
13. **…**
14. staff.isActive = !staff.isActive;
15. staff.updatedAt = new Date();
16. await staff.save();
17. return {
18. success: true,
19. message: `Staff berhasil ${staff.isActive ? 'diaktifkan' :
20. 'dinonaktifkan'}`,
21. isActive: staff.isActive,
22. };

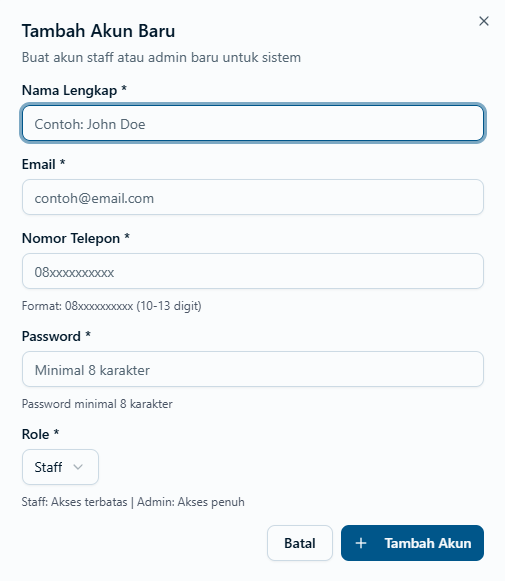
**Database yang Digunakan :**

****

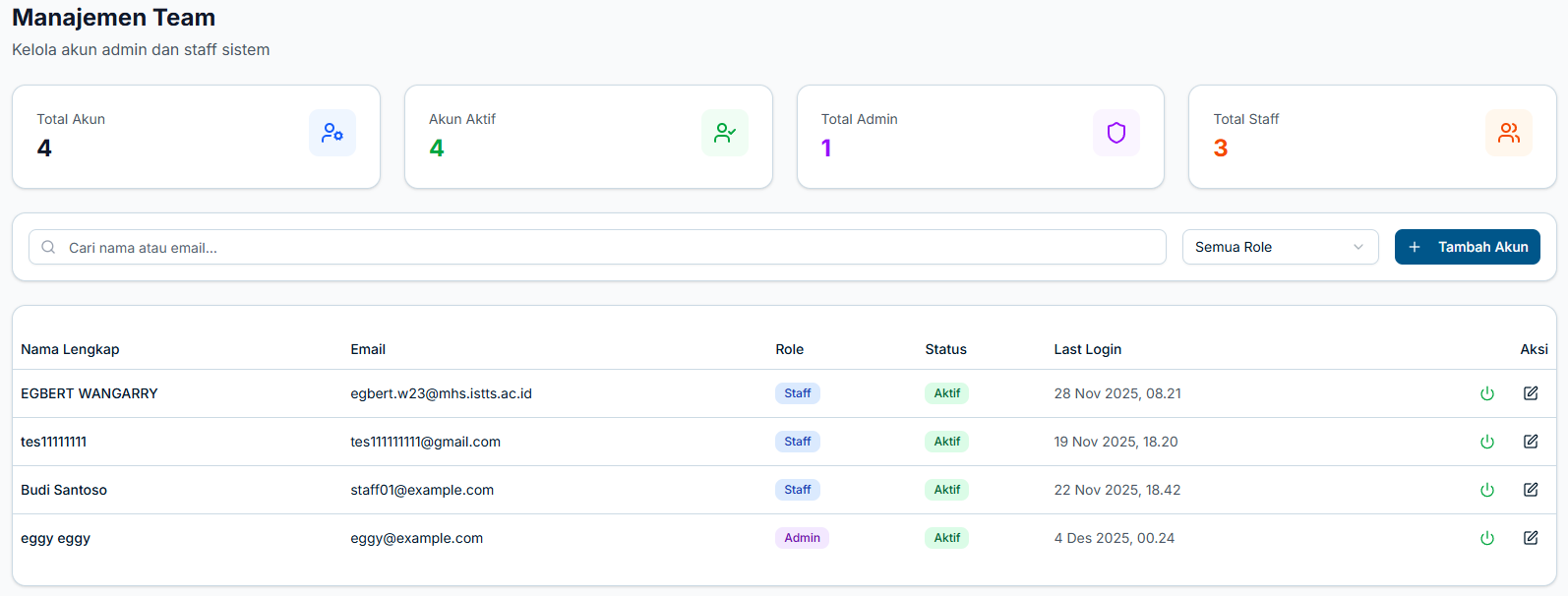
**Gambar 4.13**

**Gambar Database Akun Staff dan Admin**

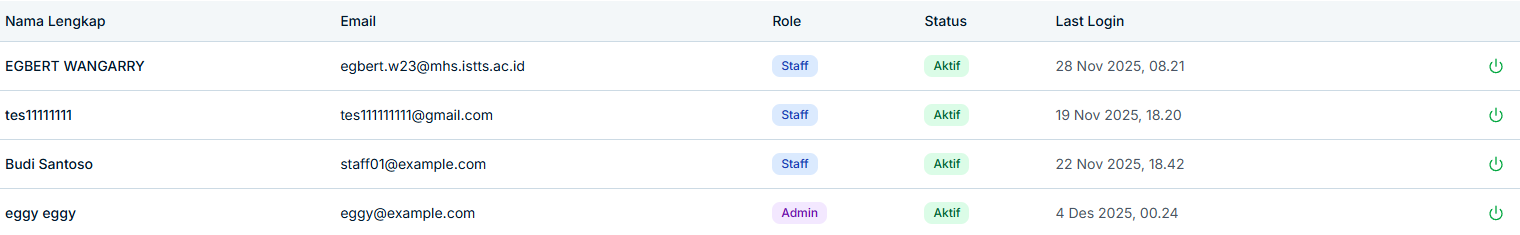
**Bentuk Gambar :**

****

**(a)**

****

**(b)**

****

**(c)**

**Gambar 4.14**

**Bentuk Gambar (a) Menambah. (b) Mengelola. (c) Menaktifasikan Akun Staff dan Admin**

**E. SP - 8 (Menambah dan Mengelola Kategori Produk dan Unit Konversi)**

Pada Algoritma 4.12, sistem memulai prosedur pembuatan kategori produk baru (01) dengan memasuki blok penanganan eksekusi (02). Langkah awal adalah verifikasi keamanan yang ketat, memastikan hanya pengguna dengan peran 'admin' yang dapat mengakses fungsi ini (03 - 06). Setelah koneksi database terjalin (07), sistem melakukan validasi duplikasi data dengan memeriksa apakah nama atau slug kategori yang diinputkan sudah ada sebelumnya. Jika ditemukan duplikasi, sistem menghentikan proses dengan pesan konflik (08-13). Untuk menjaga kerapian tampilan di frontend, sistem menghitung urutan tampilan (order) secara otomatis. Sistem mencari kategori dengan nomor urut terakhir di database, lalu menambahkan angka 1 untuk kategori baru ini, jika belum ada kategori sama sekali, urutan dimulai dari 1 (14 - 19). Selanjutnya, sistem menyusun objek kategori baru lengkap dengan data input (termasuk unit konversi), nomor urut baru, serta nilai default seperti status aktif dan jumlah produk nol (20 - 23). Objek ini disimpan ke database (24) dan sistem mengembalikan konfirmasi sukses (25). Jika terjadi kesalahan teknis, sistem menangkap error, mencatatnya, dan mengirimkan notifikasi kegagalan (27 - 28).

**Algoritma 4.12 Contoh Algoritma Menambah Kategori Produk**

1. BEGIN createCategory (input: name, slug, desc, units, etc.)
2. TRY
3. IF ctx.user.role IS NOT 'admin' THEN
4. THROW TRPCError (Code: FORBIDDEN, Message: "Only
5. admin...")
6. END IF
7. CALL connectDB()
8. GET existing FROM Database Category
9. WHERE name = input.name OR slug = input.slug
10. IF existing EXISTS THEN
11. THROW TRPCError (Code: CONFLICT, Message: "Name/Slug
12. exists")
13. END IF
14. GET lastCategory FROM Database SORT BY order DESC LIMIT 1
15. IF lastCategory EXISTS THEN
16. SET newOrder = lastCategory.order + 1
17. ELSE
18. SET newOrder = 1
19. END IF
20. SET category = NEW Category({

**Algoritma 4.12 (Lanjutan)**

1. ...input, order: newOrder,
2. isActive: TRUE, productCount: 0
3. })
4. SAVE category TO Database
5. RETURN { success: TRUE, category }
6. CATCH error
7. LOG error to Console
8. THROW TRPCError (Code: INTERNAL\_SERVER\_ERROR)
9. END

Pada Algoritma 4.13, sistem memulai prosedur pengambilan data manajemen kategori (01) dengan memasuki blok eksekusi aman (02). Langkah pertama adalah validasi hak akses, di mana sistem memastikan bahwa pengguna memiliki peran 'admin' atau 'staff'. Jika pengguna biasa (customer) mencoba mengakses data ini, permintaan akan ditolak dengan status Forbidden (03 - 06). Setelah terhubung ke database (07), sistem mengambil seluruh daftar kategori dan mengurutkannya secara otomatis berdasarkan nomor urut (order) dan abjad nama untuk kerapian tampilan (08 - 09). Sistem kemudian melakukan proses pengayaan data (data enrichment) yang krusial. Untuk setiap kategori yang ditemukan, sistem melakukan kueri tambahan ke koleksi Produk untuk menghitung berapa banyak produk yang terkait dengan kategori tersebut (10 - 13). Hal ini memberikan informasi real - time kepada admin mengenai isi setiap kategori. Selain itu, sistem juga melakukan kalkulasi statistik sederhana untuk menghitung total kategori, jumlah kategori aktif, dan jumlah kategori non - aktif (14 - 16). Paket data lengkap yang berisi daftar kategori (beserta jumlah produknya) dan ringkasan statistik kemudian dikembalikan kepada pemanggil (17). Jika terjadi kesalahan teknis, sistem menangkap error, mencatatnya, dan mengirimkan notifikasi kegagalan (19 - 20).

**Algoritma 4.13 Contoh Algoritma Mengelola Kategori Produk**

1. BEGIN getAdminAll (no input)
2. TRY
3. IF ctx.user.role NOT IN ['admin', 'staff'] THEN
4. THROW TRPCError (Code: FORBIDDEN, Message: "Only
5. admin/staff...")
6. END IF

**Algoritma 4.13 (Lanjutan)**

1. CALL connectDB()
2. GET categories FROM Database Category
3. SORT BY order ASC, name ASC
4. FOR EACH category IN categories DO
5. SET count = COUNT Documents IN Product WHERE category.name
6. ADD count TO category.productCount
7. END FOR
8. CALCULATE stats:
9. total = Count ALL, active = Count Active, inactive = Count
10. inactive
11. RETURN { categories, stats }
12. CATCH error
13. LOG error to Console
14. THROW TRPCError (Code: INTERNAL\_SERVER\_ERROR)
15. END

Pada Algoritma 4.14, sistem memulai prosedur konversi unit (01) dengan memuat konfigurasi konversi dinamis yang tersedia di sistem (02). Langkah pertama adalah mencari data rasio konversi untuk satuan asal (fromUnit) dan satuan tujuan (toUnit) dari daftar referensi (03 - 04). Sistem melakukan validasi ketersediaan data; jika salah satu satuan tidak ditemukan dalam konfigurasi, sistem mengembalikan nilai 0 untuk mencegah kesalahan perhitungan (05 - 07). Jika data valid, sistem melakukan perhitungan matematika dua tahap menggunakan konsep Base Unit (Satuan Dasar) sebagai perantara. Tahap pertama adalah menstandarisasi nilai input dengan mengalikannya terhadap rasio satuan asal untuk mendapatkan baseValue (08). Tahap kedua adalah mengonversi nilai standar tersebut ke satuan tujuan dengan membaginya terhadap rasio satuan tujuan (09). Hasil akhir dari perhitungan ini dikembalikan ke fungsi pemanggil (10). Algoritma ini memungkinkan sistem untuk memvalidasi stok secara fleksibel misalnya, memastikan apakah stok "1 Sak" (50kg) cukup untuk memenuhi pesanan "5 Kg" dengan mengonversi keduanya ke satuan yang setara sebelum dibandingkan.

**Algoritma 4.14 Contoh Algoritma Mengelola Unit Konversi**

1. BEGIN convertUnits (input: value, fromUnit, toUnit)
2. GET dynamicConversions FROM System Configuration
3. FIND fromConversion IN dynamicConversions WHERE unit = fromUnit
4. FIND toConversion IN dynamicConversions WHERE unit = toUnit

**Algoritma 4.14 (Lanjutan)**

1. IF fromConversion IS NULL OR toConversion IS NULL THEN
2. RETURN 0 (Conversion Failed)
3. END IF
4. CALCULATE baseValue = value \* fromConversion.rate
5. CALCULATE result = baseValue / toConversion.rate
6. RETURN result
7. END

Pada Segmen Program 4.12, sistem akan mencari kategori pada database berdasarkan nama dan slug yang diinputkan oleh user (02 - 04), dan sekaligus mengecek apakah kategori tersebut sudah ada didalam database atau belum, jika kategori tersebut sudah ada didalam database maka sistem akan mengirimkan pesan error “Kategori dengan nama atau slug ini sudah ada” (05 - 10). Kemudian sistem akan mencari kategori yang terakhir kali dibuat untuk memudahkan sistem membuat order baru (11 - 13). Setelah itu sistem akan membuat kategori baru berdasarkan inputan user (14 - 24), dan setelah membuat kategori baru maka sistem akan mengembalikan status success dan juga data kategori yang baru saja dibuat (25 - 29)

**Segmen Program 4.12 Contoh Program Menambah Kategori Produk**

1. …
2. const existingCategory = await Category.findOne({
3. $or: [{ name: input.name }, { slug: input.slug }],
4. });
5. if (existingCategory) {
6. throw new TRPCError({
7. code: 'CONFLICT',
8. message: 'Kategori dengan nama atau slug ini sudah ada',
9. });
10. }
11. const lastCategory = await Category.findOne().sort({ order: -1
12. }).limit(1);
13. const newOrder = lastCategory ? lastCategory.order + 1 : 1;
14. const category = await Category.create({
15. name: input.name,
16. slug: input.slug.toLowerCase(),
17. description: input.description,
18. icon: input.icon || 'folder-tree',
19. image: input.image || '',
20. availableUnits: input.availableUnits,
21. order: newOrder,
22. isActive: true,
23. productCount: 0,

**Segmen Program 4.12 (Lanjutan)**

1. });
2. return {
3. success: true,
4. category,
5. message: 'Kategori berhasil ditambahkan',
6. };

Pada Segmen Program 4.13, sistem akan mencari serta mengurutkan seluruh data kategori yang ada pada database (01 – 02). Kemudian sistem juga akan menghitung jumlah dari masing – masing kategori dari product yang memiliki kategori yang sama (03 – 13). Setelah itu sistem akan membuat total keseluruhan dari kategori yang ada dan memilah ada berapa saja kategori product yang aktif dan juga yang tidak aktif (14 – 18). Langkah terakhir sistem akan mengembalikan data kategori yang sudah diambil beserta total dari kategori, total kategori yang aktif, dan total kategori yang sudah tidak aktif (19 – 26).

**Segmen Program 4.13 Contoh Program Mengelola Kategori Produk**

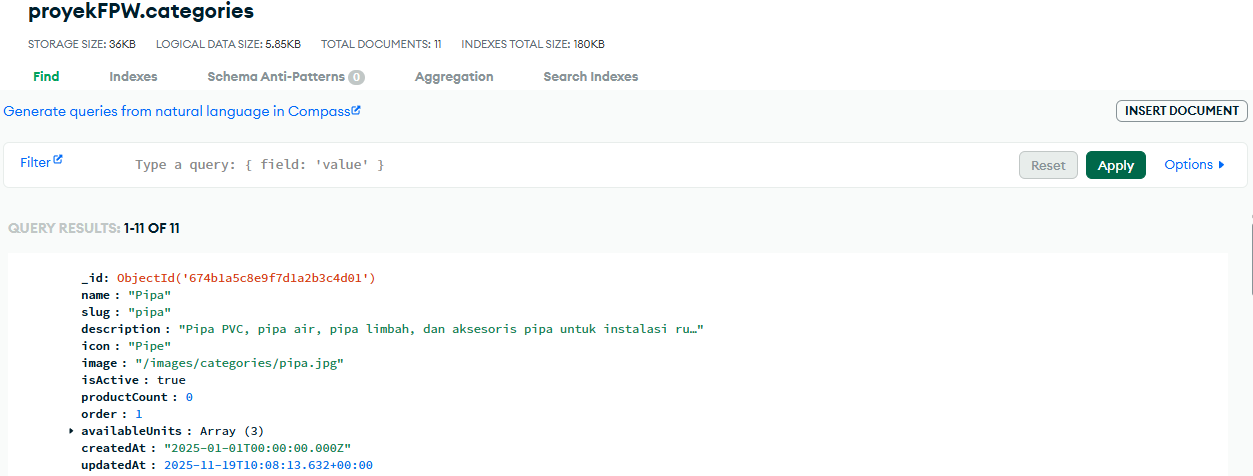
1. const categories = await Category.find().sort({ order: 1, name:
2. 1 }).lean();
3. const categoriesWithCount = await Promise.all(
4. categories.map(async (category) => {
5. const productCount = await Product.countDocuments({
6. category: category.name,
7. });
8. return {
9. ...category,
10. productCount,
11. };
12. })
13. );
14. const totalCategories = categoriesWithCount.length;
15. const activeCategories = categoriesWithCount.filter((c) =>
16. c.isActive).length;
17. const inactiveCategories = categoriesWithCount.filter((c) =>
18. !c.isActive).length;
19. return {
20. categories: categoriesWithCount,
21. stats: {
22. total: totalCategories,
23. active: activeCategories,
24. inactive: inactiveCategories,
25. },
26. };

Pada Segmen Program 4.14, sistem memiliki fungsi utilitas convertUnits yang berperan krusial dalam validasi stok dinamis. Fungsi ini didefinisikan menggunakan sintaks arrow function dengan tiga parameter, yaitu nilai nominal (value), satuan asal (from), dan satuan tujuan (to). Sistem akan mencari referensi konversi dari array dynamicConversions menggunakan metode .find() untuk mendapatkan rasio yang tepat bagi kedua satuan (02 - 07). Kemudian sistem akan melakukan Validasi null safety (08), jika salah satu satuan tidak dikenali dalam konfigurasi sistem, fungsi segera mengembalikan nilai 0 untuk mencegah kesalahan perhitungan. Kemudian sistem melakukan konversi dua tahap, yaitu menstandarisasi nilai ke satuan dasar (base unit) dengan mengalikannya terhadap rasio asal (09), kemudian membaginya dengan rasio tujuan untuk mendapatkan hasil akhir yang presisi (10) sebelum dikembalikan hasilnya (11).

**Segmen Program 4.14 Contoh Program Mengelola Unit Konversi**

1. …
2. const convertUnits = (value: number, from: string, to: string):
3. number => {
4. const fromConversion = dynamicConversions.find((c) => c.value
5. === from);
6. const toConversion = dynamicConversions.find((c) => c.value ===
7. to);
8. if (!fromConversion || !toConversion) return 0;
9. const baseValue = value \* fromConversion.conversionRate;
10. const result = baseValue / toConversion.conversionRate;
11. return result;
12. };
13. …

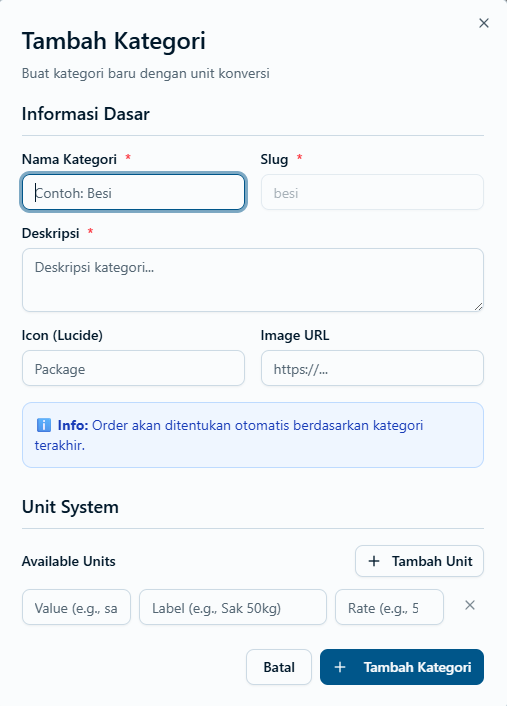
**Database yang Digunakan :**

****

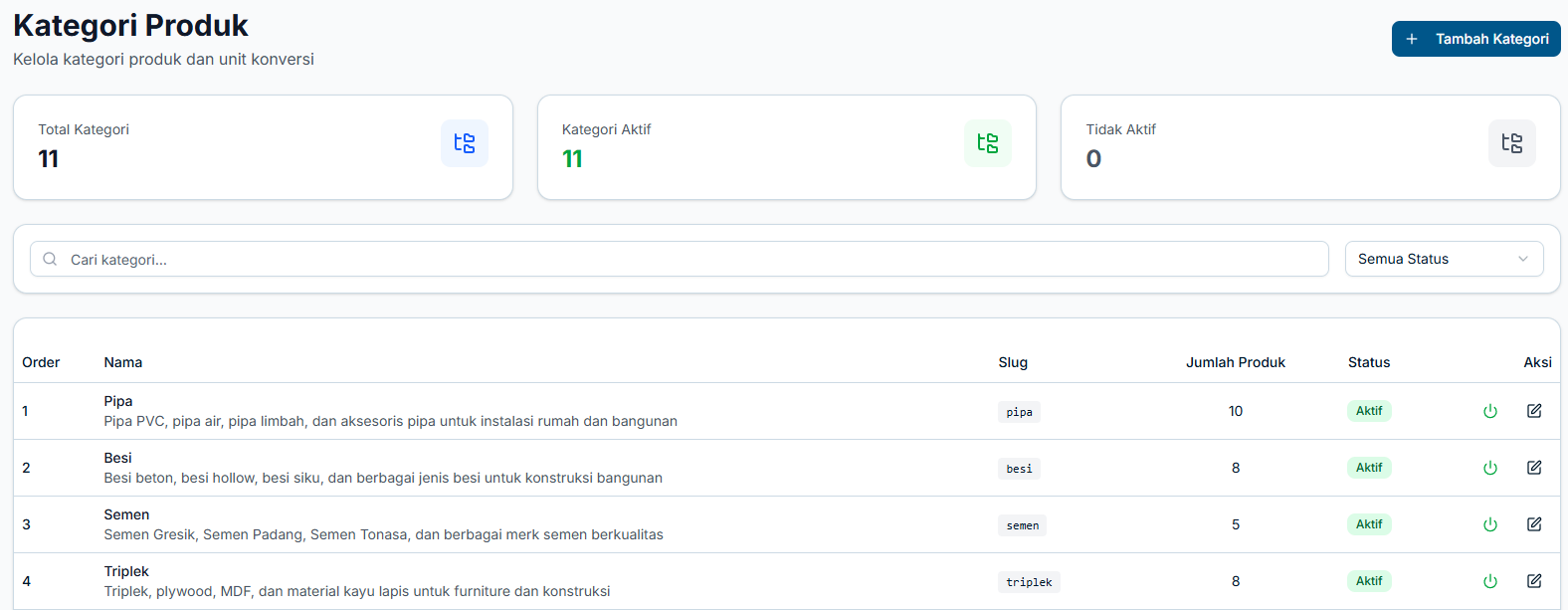
**Gambar 4.15**

**Gambar Database Kategori Produk**

**Bentuk Gambar :**

****

**(a)**

****

**(b)**

**Gambar 4.16**

**Bentuk Gambar (a) Menambah. (b) Mengelola Kategori Produk**

**4.4 Sprint Review**

Pada sesi ini, tim pengembang melakukan demonstrasi tentang bagaimana alur penggunaan program yang dibuat kepada pemiliki UD. Pelita Bangunan. Sesi ini bertujuan untuk mengumpulkan hasil kerja dan juga mendapatkan kritik/saran langsung untuk perkembangan ke sprint berikutnya.



**Gambar 4.17**

**Bukti Hasil Review Bersama Pemilik UD.**

Secara keseluruhan, pemilik toko memberikan tanggapan yang sangat positif. Beliau memberitahu bahwa fungsionalitas dasar seperti melihat laporan, pengelolaan produk, staff, dan voucher sudah berjalan dengan baik dan sesuai dengan apa yang diharapkan.

**Tabel 3.3**

**Tabel Hasil Sprint Review**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Product Backlog ID** | **Deskripsi** | **Status** |
| SR - 1 | PB - 12 | Mengelola stok produk secara real-time agar stok barang yang ditampilkan ke customer selalu akurat. | Selesai |
| SR - 2 | PB - 13 | Memproses permintaan pengembalian produk agar proses retur barang dapat tercatat dan diselesaikan sesuai prosedur. | Selesai |
| SR - 3 | PB - 14 | Menginput nomor resi pengiriman agar customer dapat melacak status pengiriman mereka. | Selesai |
| SR - 4 | PB - 15 | Menambah dan mengelola data produk agar katalog produk yang ditampilkan kepada customer selalu akurat. | Selesai |
| SR - 5 | PB - 16 | Melihat laporan transaksi, informasi inventori, dan data pengguna agar dapat membuat keputusan bisnis yang terstruktur. | Selesai |
| SR - 6 | PB - 17 | Menambah dan mengelola data voucher. | Selesai |
| SR - 7 | PB - 18 | Menambahkan, mengelola, dan mengaktifasikan akun staff dan admin. | Selesai |
| SR - 8 | PB - 19 | Menambah dan mengelola kategori produk dan unit konversi. | Selesai |

**4.5 Burndown Chart**

Burndown Chart akan terus digunakan untuk melacak kemajuan kerja tim terhadap waktu. Grafik dibawah ini akan memperlihatkan jumlah pekerjaan yang tersisa (dalam jam atau poin tugas) dari hari ke hari selama Sprint 2 berlangsung. Detail lebih lanjut mengenai tabel burndown chart dapat dilihat pada lampiran B.

Tabel burndown chart pada lampiran berfungsi melacak kemajuan kerja tim, table tersebut memecah setiap tugas dari Sprint Backlog dan melacak jam kerja actual yang dihabiskan oleh tim pengembang dalam mengembangkan program yang dibuat. Untuk total waktu yang di butuhkan untuk mengembangkan program adalah 134 jam.



**Gambar 4.18**

**Grafik Burndown Chart Sprint 2**

Contoh grafik burndown di atas menunjukkan Ideal Line (garis Orange) yang merepresentasikan laju kerja yang ideal, dan Actual Line (garis biru) yang menunjukkan sisa pekerjaan aktual. Pada Sprint 2, diharapkan Actual Line akan mendekati atau berada di bawah Ideal Line, menunjukkan efisiensi kerja tim yang terus meningkat.

**BAB V**

**UJI COBA**

Pada bab ini, dilakukan serangkaian pengujian untuk memvalidasi fungsionalitas dan kelayakan sistem informasi penjualan dan inventori UD. Pelita Bangunan. Pengujian ini bertujuan untuk memastikan bahwa seluruh alur sistem, dari sisi customer, staff, hingga admin, telah berjalan sesuai dengan kebutuhan yang telah dirumuskan pada Bab 2 dan diimplementasikan pada Bab 3 dan 4.

**5.1 Metode Pengujian**

Metode pengujian utama yang digunakan adalah Black Box Testing. Metode ini dipilih karena berfokus pada pengujian fungsionalitas sistem dari sudut pandang pengguna akhir (aktor), tanpa perlu melihat struktur kode internal. Pengujian dilakukan dengan memberikan input tertentu pada sistem dan membandingkan output (hasil aktual) dengan output yang diharapkan (hasil yang seharusnya). Sebuah fitur dinyatakan berhasil atau "Lulus" jika hasil aktual sesuai dengan hasil yang diharapkan.

**5.2 Skenario Pengujian Fungsional (Black Box)**

Untuk memastikan semua fungsionalitas teruji, skenario pengujian dirancang berdasarkan User Story yang telah didefinisikan dalam Product Backlog. Hasil pengujian didokumentasikan dalam tabel-tabel berikut, yang mencakup skenario pengujian, langkah-langkah, hasil yang diharapkan, hasil aktual, dan status kelulusan.

Tabel C.1 pada lampiran C menguraikan hasil pengujian fungsionalitas pada halaman login yang dilakukan untuk memastikan sistem keamanan dan akses pengguna berjalan sesuai rancangan. Pengujian ini mencakup sepuluh skenario uji (test cases) yang meliputi verifikasi login dengan data valid, mekanisme penanganan kesalahan (error handling) untuk input yang salah atau kosong, pengecekan status pengguna, serta pengujian fitur integrasi login menggunakan akun Google dan akses tamu.

Tabel C.2 berikut ini yang terdapat pada lampiran C menyajikan hasil pengujian fungsionalitas pada halaman registrasi yang bertujuan untuk memverifikasi keandalan proses pendaftaran akun baru. Pengujian ini meliputi skenario pendaftaran berhasil menggunakan data yang valid serta serangkaian pengujian validasi input untuk menangani kesalahan format pada kolom nama lengkap, email, username, dan nomor telepon. Selain itu, pengujian juga mencakup penerapan aturan keamanan password, seperti batasan minimal karakter dan kecocokan konfirmasi password.

Tabel 5.1 menguraikan hasil pengujian fungsionalitas pada komponen navigation bar untuk memastikan kemudahan akses pengguna ke berbagai fitur utama sistem. Pengujian ini mencakup verifikasi tautan navigasi umum seperti halaman beranda, produk, dan keranjang belanja, serta pengelolaan fitur notifikasi yang interaktif baik untuk pengguna biasa maupun admin. Selain itu, tabel ini juga menampilkan skenario pengujian pada menu profil pengguna, termasuk akses khusus ke dashboard admin, halaman riwayat pesanan, pengaturan profil, hingga fungsi logout, sebagaimana dirincikan secara lengkap dalam tabel berikut.

Tabel 5.1

**Tabel Uji Coba Navigation Bar**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Objektif | Langkah-langkah | Expected Outcome | Actual Outcome | Status |
| UJ018 | Navigasi ke halaman home. | 1. Tekan lambang “Pelita Bagunan Materials”. | Berpindah ke halaman home. | Berpindah ke halaman home. | Berhasil |
| UJ019 | Navigasi ke halaman produk. | 1. Tekan “Produk” pada navbar. | Berpindah ke halaman produk. | Berpindah ke halaman produk. | Berhasil |
| UJ020 | Navigasi ke halaman keranjang. | 1. Tekan lambang keranjang belanja. | Berpindah ke halaman keranjang. | Berpindah ke halaman keranjang. | Berhasil |

Tabel 5.1

**(Lanjutan)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Objektif | Langkah-langkah | Expected Outcome | Actual Outcome | Status |
| UJ021 | Membuka notifikasi. | 1. Tekan lambang lonceng. | Membuka notifikasi. | Membuka notifikasi. | Berhasil |
| UJ022 | Navigasi ke halaman detail pesanan. | 1. Tekan lambang lonceng. 2. Tekan isi notifikasi.. | Berpindah ke halaman detail pesanan. | Berpindah ke halaman detail pesanan. | Berhasil |
| UJ023 | Navigasi ke halaman pesanan admin. | 1. Tekan lambang lonceng. 2. Tekan isi notifikasi untuk admin | Berpindah ke halaman pesanan admin. | Berpindah ke halaman pesanan admin. | Berhasil |
| UJ024 | Menandai semua notifikasi telah dibaca. | 1. Tekan lambang lonceng. 2. Tekan “Tandai semua dibaca”. | Semua notifikasi hilang. | Semua notifikasi hilang. | Berhasil |
| UJ025 | Navigasi dashboard admin  (jika user merupakan admin atau staff). | 1. Tekan lambang profile. 2. Tekan “Dashboard Admin”. | Berpindah ke halaman dashboard admin. | Berpindah ke halaman dashboard admin. | Berhasil |
| UJ026 | Navigasi ke halaman profile. | 1. Tekan lambang profile. 2. Tekan “Profile”. | Berpindah ke halaman profile. | Berpindah ke halaman profile. | Berhasil |

Tabel 5.1

**(Lanjutan)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Objektif | Langkah-langkah | Expected Outcome | Actual Outcome | Status |
| UJ027 | Navigasi ke halaman pesanan. | 1. Tekan lambang profile. 2. Tekan “Pesanan Saya”. | Berpindah ke halaman pesanan. | Berpindah ke halaman pesanan. | Berhasil |
| UJ028 | Logout. | 1. Tekan lambang profile. 2. Tekan “Logout”. | Berpindah ke halaman login. | Berpindah ke halaman login. | Berhasil |

Tabel 5.2 menyajikan hasil pengujian terhadap fitur layanan kontak yang dirancang untuk memfasilitasi komunikasi langsung antara pengguna dan pihak pengelola. Pengujian ini berfokus pada validasi fungsi tombol pesan instan (WhatsApp) yang terintegrasi pada antarmuka pengguna, guna memastikan bahwa pintasan tersebut bekerja dengan responsif dan secara akurat mengarahkan pengguna ke aplikasi perpesanan eksternal. Rincian skenario pengujian beserta status keberhasilannya dapat dilihat pada tabel di bawah ini.

Tabel 5.2

**Tabel Uji Coba Kontak**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Objektif | Langkah-langkah | Expected Outcome | Actual Outcome | Status |
| UJ029 | Navigasi ke kontak whatsapp. | 1. Tekan tombol hijau message yang berada pada kanan bawah layar. | Berpindah ke whatsapp. | Berpindah ke whatsapp. | Berhasil |

Tabel 5.3 mendokumentasikan hasil pengujian fungsionalitas pada bagian footer situs yang berperan sebagai pusat navigasi untuk informasi pendukung dan layanan pelanggan. Fokus pengujian ini adalah memvalidasi kinerja tautan navigasi sekunder guna memastikan pengguna dapat mengakses halaman-halaman informasi penting dengan lancar. Pengujian mencakup verifikasi tautan menuju profil perusahaan, dokumen legalitas dan kebijakan, serta informasi operasional dan bantuan, sebagaimana disajikan secara rinci dalam tabel berikut.

Tabel 5.3

**Tabel Uji Coba footer**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Objektif | Langkah-langkah | Expected Outcome | Actual Outcome | Status |
| UJ030 | Navigasi ke halaman “Tentang Kami” | 1. Tekan “Tentang Kami”. | Berpindah ke halaman “Tentang Kami”. | Berpindah ke halaman “Tentang Kami”. | Berhasil |
| UJ031 | Navigasi ke halaman “Kontak”. | 1. Tekan “Kontak”. | Berpindah ke halaman “Kontak”. | Berpindah ke halaman “Kontak”. | Berhasil |
| UJ032 | Navigasi ke halaman “Syarat & Ketentuan”. | 1. Tekan “Syarat & Ketentuan”. | Berpindah ke halaman “Syarat & Ketentuan”. | Berpindah ke halaman “Syarat & Ketentuan”. | Berhasil |
| UJ033 | Navigasi ke halaman “Kebijakan Privasi”. | 1. Tekan “Kebijakan Privasi”. | Berpindah ke halaman “Kebijakan Privasi”. | Berpindah ke halaman “Kebijakan Privasi”. | Berhasil |
| UJ034 | Navigasi ke halaman “FAQ”. | 1. Tekan “FAQ”. | Berpindah ke halaman “FAQ”. | Berpindah ke halaman “FAQ”. | Berhasil |
| UJ035 | Navigasi ke halaman “Pengiriman”. | 1. Tekan “Pengiriman”. | Berpindah ke halaman “Pengiriman”. | Berpindah ke halaman “Pengiriman”. | Berhasil |
| UJ036 | Navigasi ke halaman “Pengembalian” | 1. Tekan “Pengembalian”. | Berpindah ke halaman “Pengembalian”. | Berpindah ke halaman “Pengembalian”. | Berhasil |

Tabel 5.3

**(Lanjutan)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Objektif | Langkah-langkah | Expected Outcome | Actual Outcome | Status |
| UJ037 | Navigasi ke halaman “Pembayaran”. | 1. Tekan “Pembayaran”. | Berpindah ke halaman “Pembayaran”. | Berpindah ke halaman “Pembayaran”. | Berhasil |

Tabel C.6 berikut ini yang terdapat pada lampiran C memaparkan hasil pengujian fungsionalitas pada halaman utama (Home) yang menjadi titik akses awal pengguna. Pengujian ini difokuskan pada interaktivitas elemen antarmuka seperti tombol navigasi pada jumbotron (banner utama), responsivitas fitur slider promosi, serta validasi hyperlink pada kategori produk dan daftar produk terlaris. Tujuannya adalah memastikan pengguna dapat beralih ke halaman produk atau detail promosi dengan lancar melalui berbagai elemen navigasi yang tersedia.

Tabel C.7 berikut ini yang terdapat pada lampiran C, menyajikan hasil pengujian pada halaman daftar produk (Product Page), yang dirancang untuk memudahkan pengguna menemukan barang yang diinginkan. Skenario pengujian mencakup verifikasi fungsi penyaringan (filter) berdasarkan kategori dan rentang harga, fitur pengurutan (sorting) data, serta opsi tampilan tata letak (grid atau list). Selain itu, tabel ini juga mendokumentasikan pengujian fitur pencarian, navigasi halaman (pagination), dan tombol aksi cepat untuk menambahkan produk ke keranjang.

Tabel C.8 berikut ini yang terdapat pada lampiran C, menguraikan pengujian fungsionalitas pada halaman detail produk. Fokus utama pengujian adalah memvalidasi logika interaksi pengguna terhadap produk spesifik, seperti penambahan atau pengurangan kuantitas pembelian dengan batasan stok yang tersedia, serta fungsi tombol "Tambah ke Keranjang". Selain itu, pengujian juga mencakup navigasi melalui breadcrumb, fitur berbagi tautan (share), dan akses ke rekomendasi produk terkait.

Tabel C.9 berikut ini yang terdapat pada lampiran C, menampilkan hasil uji coba pada halaman keranjang belanja (Cart). Pengujian ini bertujuan untuk memastikan sistem dapat mengelola daftar belanjaan pengguna dengan akurat, termasuk kemampuan menghapus produk, mengubah kuantitas barang dengan pembaruan subtotal secara real-time, serta validasi batas stok sebelum pengguna melanjutkan ke proses pembayaran. Navigasi untuk kembali berbelanja atau menuju halaman checkout juga turut diverifikasi.

Tabel C.10 berikut ini yang terdapat pada lampiran C, mendokumentasikan rangkaian pengujian komprehensif pada halaman checkout. Tabel ini merinci validasi formulir manajemen alamat pengiriman (termasuk penambahan alamat baru via peta atau pencarian), pemilihan metode pengiriman, serta logika penerapan kode voucher (validasi masa berlaku, kuota, dan minimal pembelian). Pengujian ini krusial untuk memastikan pengguna dapat menyelesaikan atau membatalkan pesanan tanpa hambatan teknis.

Tabel C.11 berikut ini yang terdapat pada lampiran C, berisi hasil pengujian fungsionalitas pada halaman riwayat pesanan pengguna. Skenario yang diuji meliputi penyaringan daftar pesanan berdasarkan status (seperti "Menunggu Pembayaran", "Dikirim", atau "Selesai"), fitur konfirmasi penerimaan barang, serta mekanisme pengajuan pengembalian (return) dan pemberian ulasan (rating). Tabel ini memastikan pengguna memiliki kendali penuh atas pemantauan siklus hidup pesanan mereka.

Tabel C.12 berikut ini yang terdapat pada lampiran C, menyajikan hasil pengujian pada halaman detail pesanan dengan fokus utama pada proses pembayaran. Tabel ini merinci skenario validasi input pembayaran menggunakan kartu debit (termasuk penanganan kesalahan input nomor kartu, CVV, atau token), simulasi pembayaran via QRIS, serta fitur pendukung pasca-transaksi seperti pengunduhan nota tagihan (invoice) dan opsi pemesanan ulang (repurchase).

Tabel C.13 berikut ini yang terdapat pada lampiran C, menguraikan hasil uji coba fitur manajemen akun pada halaman profil pengguna. Pengujian mencakup kemampuan pengguna untuk memperbarui informasi pribadi, mengelola buku alamat pengiriman (tambah, ubah, hapus, dan atur alamat utama), serta fitur keamanan seperti perubahan kata sandi. Validasi input yang ketat diterapkan pada setiap formulir untuk mencegah kesalahan data.

Tabel C.14 berikut ini yang terdapat pada lampiran C, memaparkan pengujian fungsionalitas pada navigation bar khusus untuk antarmuka Admin. Fokus pengujian adalah pada sistem notifikasi interaktif yang memungkinkan admin memantau aktivitas terbaru, menandai pesan telah dibaca, serta menghapus notifikasi. Selain itu, akses cepat menuju halaman toko utama dan fitur logout juga diverifikasi dalam tabel ini.

Tabel C.15 berikut ini yang terdapat pada lampiran C, menampilkan hasil uji navigasi pada sidebar menu admin. Tabel ini memastikan bahwa seluruh tautan modul manajemen—mulai dari produk, pesanan, inventori, hingga laporan—mengarahkan admin ke halaman yang tepat. Fitur responsivitas sidebar (kemampuan untuk disembunyikan atau ditampilkan kembali) juga turut diuji untuk kenyamanan tampilan antarmuka.

Tabel C.16 berikut ini yang terdapat pada lampiran C, menyajikan hasil pengujian pada dashboard utama admin. Pengujian ini berfokus pada verifikasi tombol-tombol pintasan (shortcuts) yang disediakan untuk mempercepat akses ke fungsi-fungsi krusial, seperti penambahan produk baru, pengelolaan stok, akses laporan, dan manajemen voucher, guna memastikan efisiensi kerja admin.

Tabel C.17 berikut ini yang terdapat pada lampiran C, mendokumentasikan pengujian ekstensif terkait manajemen data produk (CRUD) oleh admin. Tabel ini mencakup skenario penambahan dan pengeditan produk dengan validasi formulir yang ketat (seperti harga tidak boleh negatif, stok wajib diisi), pengelolaan atribut produk, serta fitur pencarian dan penyaringan data produk. Status aktif/nonaktif produk juga diuji untuk mengontrol visibilitas barang di sisi pengguna.

Tabel C.18 berikut ini yang terdapat pada lampiran C, menguraikan pengujian pada halaman manajemen pesanan admin. Fokus utamanya adalah kemampuan admin untuk memantau pesanan masuk melalui fitur pencarian dan filter status, serta melakukan tindakan administratif seperti membatalkan pesanan, memasukkan nomor resi pengiriman, dan memperbarui status pesanan menjadi "Terkirim".

Tabel C.19 berikut ini yang terdapat pada lampiran C, menyajikan hasil uji coba fitur pengelolaan pengembalian barang (return) di sisi admin. Pengujian mencakup alur kerja persetujuan atau penolakan pengajuan retur dari pelanggan, termasuk validasi alasan penolakan dan penyelesaian status pengembalian. Fitur pencarian dan filter status retur juga diverifikasi untuk memudahkan pemantauan.

Tabel C.20 berikut ini yang terdapat pada lampiran C, menampilkan pengujian fungsionalitas pada halaman riwayat stok (Inventory Logs). Tabel ini memverifikasi kemampuan sistem dalam menyajikan data arus keluar-masuk barang, lengkap dengan fitur pencarian produk dan penyaringan berdasarkan tipe transaksi (Stock In/Out) serta periode waktu tertentu (harian, mingguan, bulanan).

Tabel C.21 berikut ini yang terdapat pada lampiran C, berisi hasil pengujian manajemen data pelanggan. Skenario mencakup fitur pencarian akun pelanggan, penyaringan berdasarkan status, serta tindakan administratif untuk menonaktifkan atau mengaktifkan kembali akun pengguna. Integrasi tombol pintasan WhatsApp untuk komunikasi langsung dengan pelanggan juga turut diuji.

Tabel C.22 berikut ini yang terdapat pada lampiran C, mendokumentasikan pengujian fitur manajemen tim internal (Admin dan Staff). Tabel ini merinci proses penambahan akun baru dengan validasi format data (email, nomor telepon, password), pengeditan peran pengguna (role), serta pengelolaan status aktif akun tim. Hal ini penting untuk mengatur hak akses dalam sistem backend.

Tabel C.23 berikut ini yang terdapat pada lampiran C, menguraikan pengujian pada halaman manajemen kategori produk. Pengujian difokuskan pada validasi formulir saat admin menambah atau mengubah kategori dan unit satuan ukur, memastikan integritas data master produk. Selain itu, fitur pencarian dan pengaturan status aktif kategori juga diverifikasi fungsinya.

Tabel C.24 berikut ini yang terdapat pada lampiran C, menyajikan hasil uji coba pengelolaan voucher promosi. Tabel ini menyoroti validasi logika bisnis yang kompleks saat pembuatan voucher, seperti pengecekan rentang tanggal berlaku, batas penggunaan, dan nilai diskon (nominal atau persentase). Fitur pencarian dan perubahan status voucher juga diuji untuk memastikan promosi berjalan sesuai rencana.

Tabel C.25 berikut ini yang terdapat pada lampiran C, menampilkan pengujian fitur pelaporan dan analitik sistem. Tabel ini mencakup verifikasi akses terhadap berbagai jenis laporan statistik (penjualan, stok, pelanggan), penggunaan filter periode waktu kustom, serta visualisasi data dalam bentuk grafik (chart). Fungsi ekspor data ke format PDF dan Excel juga diuji untuk memastikan admin dapat mengunduh arsip laporan dengan benar.

**BAB VI**

**PENUTUP**

Pengembangan perangkat lunak yang telah dilakukan merupakan upaya untuk menciptakan solusi teknologi yang dapat meningkatkan efisiensi dan efektivitas dalam pengelolaan data usaha dagang. Melalui proses analisis, perancangan, dan pengujian yang komprehensif, sistem ini dirancang untuk menggantikan sistem manual yang rentan terhadap kesalahan dengan sistem yang lebih terstruktur, terintegrasi, dan mudah dikelola. Dengan memanfaatkan teknologi modern, sistem ini diharapkan dapat memberikan dampak positif bagi operasional usaha dagang dan mendukung pertumbuhan bisnis secara berkelanjutan.

1. **Kesimpulan**

Berdasarkan hasil analisis, perancangan, dan implementasi yang telah dilakukan, pengembangan Sistem Informasi Penjualan dan Inventori pada UD. Pelita Bangunan telah berhasil diselesaikan dengan baik. Sistem berbasis website yang dibangun mampu menjalankan seluruh alur kerja operasional antara Customer, Staff, dan Admin dalam satu platform terpusat. Keberadaan sistem ini secara efektif menggantikan proses pencatatan manual yang sebelumnya rentan terhadap kesalahan, sehingga pengelolaan data bisnis menjadi lebih terstruktur, dan aman.

Dari sisi fungsionalitas, sistem telah mampu menangani kompleksitas pengelolaan data bahan bangunan melalui fitur Manajemen Inventori yang dilengkapi dengan fitur Konversi Unit Produk. Fitur ini mampu membuat perhitungan stok seperti konversi satuan Sak ke Kilogram tetap terjaga secara real-time saat transaksi terjadi. Selain itu, efisiensi operasional juga meningkat dengan adanya proses checkout yang terintegrasi dengan kalkulasi ongkos kirim otomatis, serta fitur Redirect WhatsApp yang mengatasi masalah komunikasi antara pelanggan dan staf toko.

Secara teknis, penggunaan teknologi modern seperti React.js dan tRPC terbukti memberikan performa sistem yang responsif, didukung oleh fitur Notifikasi Inbox yang memberikan transparansi status pesanan kepada pengguna. Validitas sistem ini juga telah dikonfirmasi melalui rangkaian Uji Coba (Black Box Testing) pada Bab 5, di mana seluruh fitur utama maupun pendukung dinyatakan telah berjalan sesuai dengan tujuan yang dirancang. Dengan demikian, sistem ini dinyatakan siap digunakan untuk mendukung keberlangsungan serta pengembangan bisnis UD. Pelita Bangunan.

1. **Saran**

Meskipun sistem ini telah memenuhi kebutuhan dasar dan fungsionalitas utama UD. Pelita Bangunan, tetap saja sistem ini masih terdapat ruang untuk perbaikan dan pengembangan. Berikut adalah beberapa saran yang dapat dipertimbangkan untuk pengembangan sistem selanjutnya:

1. Pengembangan versi aplikasi mobile dapat dipertimbangkan untuk memberikan akses yang lebih mudah bagi customer, serta fitur push notification yang lebih interaktif dibandingkan notifikasi berbasis web.
2. Saat ini pelacakan dilakukan berbasis input nomor resi manual. Kedepannya, dapat ditambahkan fitur integrasi langsung untuk menampilkan posisi kurir secara real-time pada website.