MATA KULIAH PEMROGRAMAN BERBASIS KERANGKA KERJA

RANCANG BANGUN SISTEM E-COMMERCE "BELANJAIN" BERBASIS WEB MENGGUNAKAN ARSITEKTUR FULL-STACK (NESTJS & REACT)

DISUSUN OLEH:

Zabrina Zerlynda Musanef

(5025231267)

DOSEN:

Moch. Nafkhan Alzamzami, S.T., M.T.

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

Semester Gasal 2025/2026

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi yang sangat pesat dalam satu dekade terakhir telah mengubah paradigma perdagangan konvensional menjadi perdagangan digital atau yang dikenal sebagai *Electronic Commerce* (E-commerce). Transformasi ini menuntut para pelaku usaha, mulai dari skala mikro hingga menengah, untuk beradaptasi dengan platform digital guna menjangkau pasar yang lebih luas dan meningkatkan efisiensi operasional. Namun, banyak solusi *marketplace* yang ada saat ini sering kali memiliki antarmuka yang terlalu kompleks bagi penjual pemula atau sebaliknya, terlalu sederhana sehingga tidak menyediakan fitur analitik yang memadai untuk pengambilan keputusan bisnis. Kebutuhan akan sebuah sistem yang tidak hanya memfasilitasi transaksi jual-beli, tetapi juga memberikan kendali penuh kepada penjual atas inventaris dan performa toko mereka, menjadi sangat krusial. Oleh karena itu, pengembangan aplikasi web yang responsif, aman, dan kaya fitur menjadi solusi yang relevan untuk menjawab tantangan tersebut.

Dalam ekosistem *marketplace* modern, tantangan teknis terbesar sering kali terletak pada pengelolaan transaksi yang melibatkan banyak penjual (*multi-vendor*) dalam satu kali proses *checkout*. Mekanisme ini, yang dikenal sebagai *Split Order*, memerlukan logika *backend* yang canggih untuk memisahkan pesanan berdasarkan toko, menghitung total pembayaran secara akurat, dan mendistribusikan notifikasi status pesanan kepada masing-masing pihak yang terlibat. Selain itu, aspek keamanan data pengguna, manajemen inventaris yang *real-time*, serta kemampuan sistem untuk memberikan laporan penjualan otomatis (*automated reporting*) menjadi indikator utama kualitas sebuah platform e-commerce. Tanpa sistem yang terintegrasi dengan baik, penjual akan kesulitan memantau stok yang menipis atau menganalisis tren penjualan harian mereka, yang pada akhirnya dapat menghambat pertumbuhan bisnis.

Berdasarkan permasalahan dan kebutuhan tersebut, proyek ini bertujuan untuk merancang dan membangun aplikasi E-commerce bernama "Belanjain". Aplikasi ini dikembangkan dengan arsitektur *Full-Stack* modern, memisahkan sisi *frontend* yang interaktif menggunakan React dan sisi *backend* yang tangguh menggunakan NestJS. Proyek ini tidak hanya berfokus pada fitur standar CRUD (*Create, Read, Update, Delete*), tetapi juga mengimplementasikan fitur tingkat lanjut seperti dasbor analitik visual, otomatisasi laporan mingguan menggunakan *Cron Job*, sistem keamanan berbasis *Role-Based Access Control* (RBAC), serta manajemen transaksi yang kompleks. Melalui proyek ini, diharapkan tercipta sebuah platform simulasi *marketplace* yang handal, aman, dan memberikan pengalaman pengguna (*User Experience*) yang optimal bagi penjual maupun pembeli.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan di atas, maka rumusan masalah yang akan diselesaikan dalam proyek akhir ini adalah sebagai berikut:

1. Bagaimana merancang dan mengimplementasikan arsitektur sistem *Full-Stack* yang aman dan terukur menggunakan NestJS dan React untuk memfasilitasi interaksi antara penjual dan pembeli?
2. Bagaimana membangun logika *backend* yang mampu menangani transaksi kompleks, khususnya fitur *Split Order*, di mana satu keranjang belanja dapat berisi produk dari berbagai toko yang berbeda dan menghasilkan nomor pesanan yang terpisah?
3. Bagaimana mengimplementasikan fitur manajemen inventaris yang cerdas, termasuk sistem peringatan dini (*alert system*) untuk stok produk yang menipis dan visualisasi data penjualan (grafik tren) untuk membantu penjual dalam pengambilan keputusan?
4. Bagaimana menerapkan mekanisme otomatisasi sistem, seperti pengiriman notifikasi email transaksional dan laporan penjualan mingguan secara terjadwal (*Cron Jobs*), guna meningkatkan efisiensi operasional pengguna?

1.3 Tujuan

Tujuan utama dari pengembangan sistem aplikasi E-commerce "Belanjain" ini adalah:

1. Menghasilkan aplikasi web E-commerce berbasis *Full-Stack* yang memenuhi standar keamanan industri, termasuk implementasi autentikasi JWT (*JSON Web Token*) dengan mekanisme *Refresh Token* dan enkripsi kata sandi yang aman.
2. Mengimplementasikan fitur *Split Order* pada sistem *checkout* yang memungkinkan sistem memecah pesanan secara otomatis berdasarkan identitas penjual, sehingga memudahkan manajemen status pesanan di sisi masing-masing toko.
3. Menyediakan dasbor khusus penjual (*Seller Dashboard*) yang informatif dan interaktif, mencakup fitur manajemen produk, pemantauan stok dengan indikator visual, serta grafik analitik pendapatan harian, mingguan, dan bulanan.
4. Membangun layanan latar belakang (*background service*) yang andal untuk menangani tugas-tugas otomatis, seperti pengiriman email konfirmasi pesanan, notifikasi stok rendah, dan rekapitulasi laporan penjualan mingguan kepada penjual.

1.4 Batasan Proyek

Agar pengerjaan proyek ini lebih terarah dan fokus pada tujuan utama, maka ditetapkan beberapa batasan masalah sebagai berikut:

1. Aplikasi ini berbasis web (*website*) dan dioptimalkan untuk penggunaan melalui peramban (*browser*) desktop, meskipun tetap responsif untuk perangkat *mobile*.
2. Sistem pembayaran (*Payment Gateway*) dan perhitungan ongkos kirim (*Shipping Cost*) bersifat simulasi. Tidak ada integrasi dengan layanan perbankan nyata atau API logistik pihak ketiga (seperti RajaOngkir) untuk menyederhanakan ruang lingkup pengembangan pada logika inti aplikasi.
3. Fitur notifikasi difokuskan pada pengiriman melalui Email menggunakan protokol SMTP (Gmail), tidak mencakup notifikasi via SMS atau WhatsApp.
4. Peran pengguna dalam sistem dibatasi pada dua peran utama, yaitu Customer (Pembeli) dan Seller (Penjual). Tidak ada panel khusus *Super Admin* untuk

pengelolaan seluruh sistem, karena fokus proyek adalah pada interaksi jual-beli dan manajemen toko mandiri.

BAB II
ANALISIS KEBUTUHAN SISTEM

2.1 Kebutuhan Fungsional (Functional Requirements)

Analisis kebutuhan fungsional mendefinisikan layanan-layanan yang harus disediakan oleh sistem, bagaimana sistem bereaksi terhadap input tertentu, dan perilaku sistem dalam situasi tertentu. Berdasarkan tujuan pengembangan aplikasi "Belanjain", kebutuhan fungsional sistem ini dikelompokkan menjadi beberapa modul utama yang mencakup seluruh alur proses bisnis E-commerce, mulai dari manajemen akun hingga pelaporan otomatis.

1. Modul Manajemen Pengguna dan Keamanan (User Management & Security)
   Sistem harus memfasilitasi proses registrasi dan autentikasi yang aman bagi pengguna. Calon pengguna, baik Penjual (*Seller*) maupun Pembeli (*Customer*), diwajibkan melakukan pendaftaran dengan menyertakan data valid seperti nama, email, dan kata sandi yang memenuhi kriteria keamanan. Sistem mengimplementasikan mekanisme *Login* menggunakan standar JWT (*JSON Web Token*) untuk menghasilkan token akses (*Access Token*) dan token penyegar (*Refresh Token*), yang memungkinkan pengguna tetap masuk dalam sesi yang panjang tanpa mengurangi aspek keamanan. Selain itu, terdapat fitur manajemen profil yang memungkinkan pengguna untuk memperbarui informasi pribadi, mengubah kata sandi, serta bagi penjual, mengatur informasi toko (Nama Toko dan Lokasi) untuk keperluan *branding* dan logistik.
2. Modul Manajemen Produk dan Inventaris (Product & Inventory Management)
   Sebagai inti dari aktivitas penjualan, sistem menyediakan fitur pengelolaan produk yang komprehensif bagi penjual. Penjual dapat menambahkan produk baru dengan atribut lengkap meliputi nama, deskripsi, harga, kategori, serta mengunggah gambar produk (*image upload*) dan menetapkan varian (seperti warna atau ukuran). Untuk meningkatkan efisiensi operasional, sistem menyediakan halaman Inventaris Khusus (*Dedicated Inventory Page*) yang memungkinkan penjual memantau dan memperbarui jumlah stok secara cepat (*quick update*) tanpa harus masuk ke halaman detail setiap produk. Sistem juga dilengkapi dengan logika peringatan dini (*Low Stock Alerts*), di mana produk dengan stok di bawah 10 unit akan ditandai secara visual untuk mencegah kekosongan barang.
3. Modul Pencarian dan Transaksi (Browsing & Transaction)
   Sistem dirancang untuk memberikan pengalaman belanja yang intuitif bagi pembeli. Fitur pencarian dan penyaringan (*Search & Filter*) memungkinkan pembeli menemukan produk berdasarkan kata kunci, kategori, rentang harga, hingga tingkat popularitas. Produk yang dipilih dapat dimasukkan ke dalam Keranjang Belanja (*Shopping Cart*), di mana sistem secara otomatis mengelompokkan barang berdasarkan toko penjualnya. Fitur unggulan dalam modul ini adalah mekanisme *Split Order* pada saat *Checkout*. Jika pembeli melakukan transaksi untuk produk dari beberapa toko yang berbeda dalam satu kali pembayaran, sistem secara cerdas akan memecah pesanan tersebut menjadi

beberapa Nomor Order (Order ID) yang terpisah, memastikan setiap penjual hanya menerima data pesanan yang relevan dengan toko mereka.

4. Modul Manajemen Pesanan (Order Management)
   Sistem menyediakan antarmuka terpisah bagi penjual dan pembeli untuk memantau siklus hidup pesanan. Di sisi penjual, terdapat fitur "Kelola Pesanan" (*Manage Orders*) yang memungkinkan penjual melihat pesanan masuk dan memperbarui status pesanan secara bertahap, mulai dari *Processing*, *Shipped*, hingga *Delivered*. Di sisi lain, pembeli memiliki akses ke halaman "Pesanan Saya" (*My Orders*) untuk melacak status barang, membatalkan pesanan yang masih berstatus *Pending*, serta melakukan konfirmasi penerimaan barang dan memberikan ulasan (*Review*) ketika pesanan telah selesai.
5. Modul Analitik dan Otomatisasi (Analytics & Automation)
   Untuk mendukung pengambilan keputusan bisnis, sistem menyediakan Dasbor Analitik (*Sales Dashboard*) bagi penjual yang menyajikan data visual berupa grafik tren penjualan harian, ringkasan pendapatan (harian, mingguan, bulanan), serta daftar produk terlaris. Selain itu, sistem mengimplementasikan layanan otomatisasi latar belakang (*Background Service*) menggunakan *Cron Job* yang berjalan secara periodik. Layanan ini bertugas mengirimkan laporan penjualan mingguan (*Weekly Sales Report*) secara otomatis ke email penjual. Seluruh aktivitas krusial seperti konfirmasi pesanan baru, perubahan status pengiriman, dan peringatan stok menipis juga terintegrasi dengan layanan notifikasi email (*Email Notification*) secara *real-time*.

2.2 Kebutuhan Non-Fungsional (Non-Functional Requirements)

Kebutuhan non-fungsional mendefinisikan batasan layanan atau fungsi sistem, seperti kendala waktu, batasan proses pengembangan, dan standar keamanan yang harus dipatuhi.

1. Keamanan (*Security*): Sistem harus menjamin kerahasiaan data pengguna. Seluruh kata sandi wajib disimpan dalam bentuk terenkripsi menggunakan algoritma *hashing* yang kuat (Bcrypt). Akses ke fitur-fitur sensitif (seperti manajemen toko) harus dibatasi menggunakan mekanisme *Role-Based Access Control* (RBAC) untuk memastikan hanya pengguna dengan peran yang tepat yang dapat mengaksesnya.
2. Kinerja (*Performance*): Sistem harus mampu menangani permintaan data dalam jumlah besar dengan waktu respons yang cepat. Hal ini dicapai melalui penerapan teknik *Pagination* pada halaman daftar produk dan riwayat pesanan untuk mengurangi beban muat data, serta pengindeksan (*indexing*) pada basis data untuk mempercepat proses pencarian.
3. Ketersediaan (*Availability*): Sistem dirancang untuk dapat diakses kapan saja melalui peramban web standar. Arsitektur *backend* yang terpisah (*Decoupled Architecture*) memungkinkan layanan API tetap berjalan independen meskipun antarmuka pengguna sedang dalam pemeliharaan, atau sebaliknya.
4. Antarmuka Pengguna (*Usability*): Antarmuka aplikasi harus responsif (*Mobile-Responsive*), menyesuaikan tata letak secara otomatis baik pada layar desktop maupun perangkat seluler, serta memberikan umpan balik visual (*User Feedback*)

yang jelas seperti notifikasi sukses atau pesan kesalahan pada setiap interaksi pengguna.

2.3 Spesifikasi Teknis (System Specifications)

Berdasarkan analisis kebutuhan di atas, berikut adalah matriks spesifikasi teknis utama yang diimplementasikan dalam pengembangan sistem "Belanjain":

No	Kategori Fitur	Deskripsi Implementasi Teknis
1	CRUD RESTful API	Implementasi operasi *Create, Read, Update, Delete* pada entitas Produk, Pesanan, dan Keranjang menggunakan protokol HTTP standar.
2	Autentikasi	Mekanisme Login/Register menggunakan JWT (*JSON Web Token*) dengan strategi validasi ganda (*Access Token* & *Refresh Token*).
3	File Upload	Fitur unggah gambar produk dengan validasi tipe file (JPG/PNG) dan batasan ukuran file (maksimal 5MB) menggunakan pustaka Multer.
4	Email Service	Layanan pengiriman email transaksional menggunakan protokol SMTP (Nodemailer) untuk notifikasi pesanan dan laporan mingguan.
5	Search & Filtering	Logika pencarian dinamis pada sisi *backend* yang mendukung filter berdasarkan kategori, rentang harga, dan pengurutan data (*Sorting*).
6	Database Transaction	Penggunaan *Prisma Transactions* untuk menjamin integritas data saat proses *Checkout* (pembuatan pesanan sekaligus pengurangan stok dan penghapusan keranjang).
7	Task Scheduling	Implementasi penjadwalan tugas otomatis (*Cron*) untuk memicu pengiriman laporan rekapitulasi penjualan setiap minggu tanpa intervensi manual

BAB III
PERANCANGAN SISTEM

Tahapan perancangan sistem merupakan fase krusial dalam pengembangan perangkat lunak yang bertujuan untuk menerjemahkan kebutuhan fungsional dan non-fungsional ke dalam representasi teknis yang siap diimplementasikan. Pada bab ini, akan dijelaskan secara rinci mengenai arsitektur sistem yang dibangun, perancangan basis data yang digunakan untuk menyimpan informasi, serta struktur kode proyek yang menerapkan prinsip modularitas.

3.1 Arsitektur Aplikasi

Sistem E-commerce "Belanjain" dikembangkan dengan menggunakan pendekatan arsitektur terpisah (*Decoupled Architecture*) atau sering disebut sebagai pola *Client-Server* modern. Dalam arsitektur ini, sisi antarmuka pengguna (*Frontend*) dan sisi logika bisnis (*Backend*) beroperasi sebagai dua entitas independen yang berkomunikasi melalui protokol HTTP standar.

1. Sisi Klien (Frontend - React): Bertanggung jawab penuh atas interaksi pengguna dan visualisasi data. Dikembangkan menggunakan pustaka React dengan dukungan Tailwind CSS untuk penataan gaya (*styling*) yang responsif. Sisi klien menerapkan konsep *Single Page Application* (SPA), di mana perpindahan halaman terjadi secara dinamis tanpa perlu memuat ulang seluruh halaman peramban, memberikan pengalaman pengguna yang cepat dan mulus. Komunikasi data ke server dilakukan menggunakan pustaka Axios yang dikonfigurasi dengan *Interceptor* untuk menyisipkan Token Autentikasi (JWT) secara otomatis pada setiap permintaan (*request*).
2. Sisi Server (Backend - NestJS): Bertindak sebagai penyedia layanan API (*Application Programming Interface*). Dikembangkan menggunakan kerangka kerja NestJS yang berjalan di atas lingkungan Node.js. NestJS dipilih karena arsitekturnya yang modular dan terstruktur, serta dukungannya terhadap TypeScript yang menjamin keamanan tipe data (*type safety*). Backend ini menangani seluruh logika bisnis yang kompleks, seperti validasi input, enkripsi kata sandi, perhitungan total transaksi, penjadwalan tugas otomatis (*Task Scheduling*), serta interaksi dengan basis data.
3. Lapisan Basis Data (Database Layer): Sistem menggunakan MySQL sebagai sistem manajemen basis data relasional (RDBMS) untuk menyimpan data persisten. Interaksi antara kode aplikasi (Backend) dan database dijembatani oleh Prisma ORM (*Object-Relational Mapping*). Penggunaan Prisma memungkinkan manipulasi data dilakukan menggunakan kode TypeScript yang intuitif tanpa perlu menulis kueri SQL mentah secara manual, sekaligus meminimalkan risiko kesalahan sintaksis dan serangan injeksi SQL (*SQL Injection*).

3.2 Perancangan Basis Data (Entity Relationship Diagram)

Perancangan basis data dilakukan dengan memodelkan entitas-entitas utama yang terlibat dalam proses bisnis E-commerce. Skema database didefinisikan menggunakan

Prisma Schema Language yang kemudian dimigrasikan ke dalam tabel-tabel MySQL. Berikut adalah entitas utama dalam sistem:

1. User (Pengguna): Menyimpan data otentikasi dan profil. Tabel ini memiliki kolom role (ENUM) untuk membedakan antara 'SELLER' dan 'CUSTOMER'. Terdapat relasi *One-to-Many* ke tabel Product (jika user adalah penjual) dan ke tabel Order (jika user adalah pembeli).
2. Product (Produk): Menyimpan informasi barang dagangan. Tabel ini memiliki fitur unik berupa kolom variants dan specifications yang menggunakan tipe data JSON. Hal ini memungkinkan penyimpanan struktur data yang fleksibel (seperti varian warna atau ukuran yang dinamis) tanpa perlu membuat banyak tabel tambahan yang kompleks.
3. Cart & CartItem (Keranjang): Tabel Cart terhubung *One-to-One* dengan User, sedangkan CartItem menyimpan daftar produk yang dipilih beserta kuantitas dan varian spesifik yang dipilih pembeli. Data varian yang dipilih juga disimpan dalam format JSON untuk fleksibilitas.
4. Order & OrderItem (Pesanan): Tabel Order merepresentasikan satu transaksi final. Tabel ini mencatat informasi krusial seperti totalPrice, status pengiriman, dan shippingAddress. Tabel OrderItem berfungsi sebagai tabel perantara (*junction table*) yang mencatat detil produk apa saja yang ada dalam satu pesanan, termasuk harga saat transaksi terjadi (*snapshot price*), guna menjaga integritas data laporan keuangan meskipun harga produk asli berubah di masa depan.
5. Review (Ulasan): Tabel yang menghubungkan User dan Product, memungkinkan pembeli memberikan rating dan komentar setelah transaksi selesai.

Relasi antar tabel dirancang untuk mendukung integritas referensial, di mana penghapusan data induk (misalnya User) akan ditangani dengan kebijakan yang s

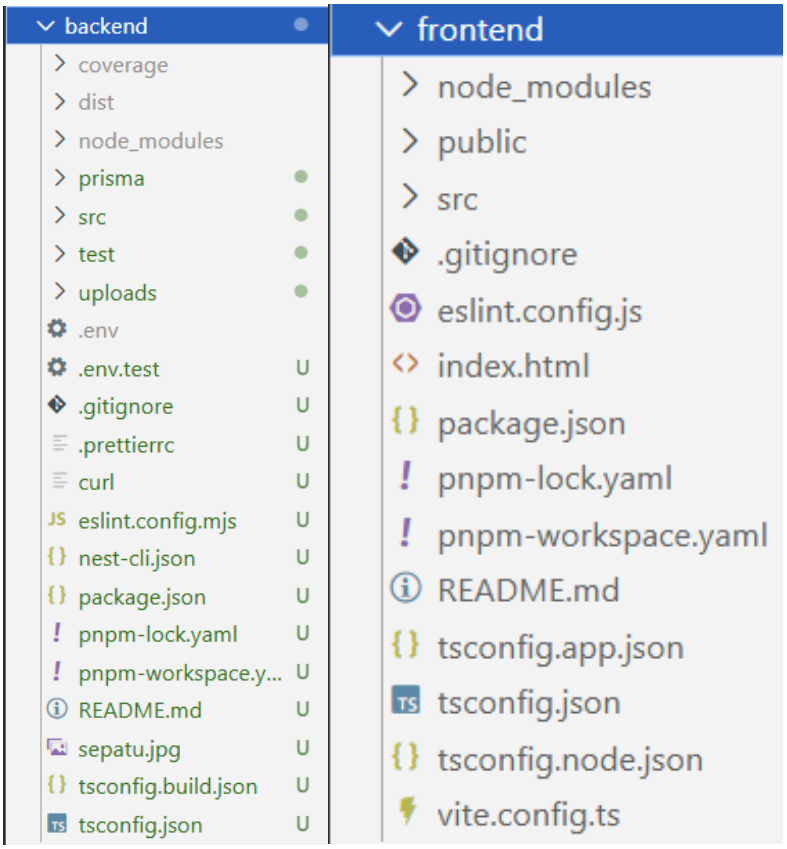
3.3 Struktur Proyek (Project Structure)

Penerapan struktur kode yang rapi dan terorganisir sangat penting untuk mempermudah pemeliharaan dan pengembangan fitur di masa depan. Proyek ini mengadopsi pola *Modular Monolith* pada sisi Backend dan *Feature-based Folder Structure* pada sisi Frontend.

1. Struktur Backend (NestJS): Kode dikelompokkan berdasarkan modul fitur. Setiap folder modul (misalnya src/products/ atau src/orders/) memiliki komponen lengkapnya sendiri yang terdiri dari:
    - Controller (*.controller.ts): Bertugas menerima permintaan HTTP dari klien dan mengirimkan respons.
    - Service (*.service.ts): Berisi logika bisnis utama dan algoritma pemrosesan data.
    - DTO (*.dto.ts): *Data Transfer Object* yang mendefinisikan bentuk dan validasi data yang dikirim oleh klien.
    - Module (*.module.ts): Berkas konfigurasi yang menyatukan controller dan service serta mengatur dependensi antar modul (misalnya modul Dashboard mengimpor modul Email).

2. Struktur Frontend (React): Kode sisi klien diorganisir untuk memisahkan antara tampilan halaman dan komponen yang dapat digunakan kembali (*reusable components*).
    - src/pages/: Berisi komponen halaman utama yang sesuai dengan rute aplikasi (misalnya Dashboard.tsx, MyInventory.tsx, SellerAnalytics.tsx).
    - src/components/: Berisi elemen antarmuka kecil yang digunakan di banyak halaman, seperti Navbar, Layout, dan ProductCard.
    - src/api/: Berisi konfigurasi axiosInstance untuk sentralisasi komunikasi ke backend.

Pendekatan struktur ini memastikan bahwa setiap bagian kode memiliki tanggung jawab tunggal (*Single Responsibility Principle*), sehingga kode menjadi lebih mudah dibaca, diuji (*testable*), dan dikembangkan lebih lanjut.



BAB IV
IMPLEMENTASI FITUR UTAMA

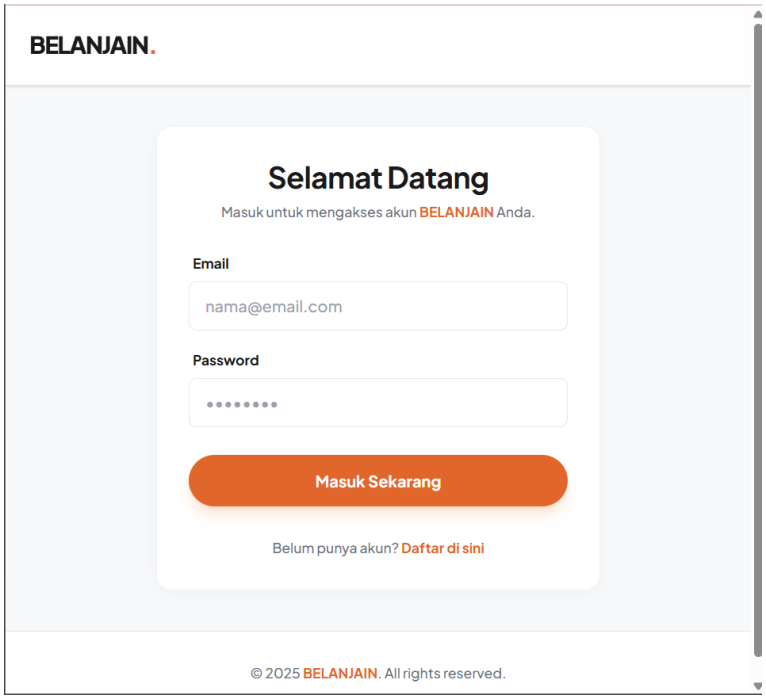
Bab ini membahas realisasi dari rancangan sistem yang telah didefinisikan sebelumnya. Fokus utama bab ini adalah menjabarkan bagaimana logika pemrograman diterapkan untuk memenuhi kebutuhan fungsional, mulai dari mekanisme keamanan, manajemen data produk yang dinamis, hingga algoritma transaksi kompleks yang memisahkan pesanan berdasarkan penjual.

4.1 Implementasi Keamanan dan Autentikasi

Keamanan merupakan fondasi utama dalam aplikasi "Belanjain". Implementasi autentikasi dibangun menggunakan standar industri JWT (*JSON Web Token*) untuk menjamin keamanan pertukaran data antara klien dan server tanpa membebani server dengan penyimpanan sesi (*stateless*).

Pada sisi *Backend* (NestJS), modul autentikasi (AuthModule) menangani dua proses krusial: registrasi dan *login*. Saat pengguna melakukan registrasi, sistem tidak menyimpan kata sandi dalam bentuk teks asli (*plain text*). Menggunakan pustaka Bcrypt, kata sandi dienkripsi melalui proses *hashing* dengan *salt round* tertentu sebelum disimpan ke dalam basis data. Hal ini memitigasi risiko kebocoran data kredensial jika basis data diretas. Saat pengguna melakukan *login*, sistem memverifikasi kredensial dan menerbitkan dua jenis token: *Access Token* (berlaku singkat untuk otorisasi API) dan *Refresh Token* (berlaku panjang untuk memperbarui sesi). Selain itu, diterapkan mekanisme *Role-Based Access Control* (RBAC) menggunakan *Guards* dan *Decorators* kustom (@Roles). Mekanisme ini memastikan bahwa *endpoint* sensitif, seperti manajemen inventaris, hanya dapat diakses oleh pengguna dengan peran 'SELLER', dan akan menolak akses dari 'CUSTOMER' dengan kode status 403 Forbidden.

Pada sisi *Frontend* (React), keamanan diimplementasikan melalui komponen ProtectedRoute. Komponen ini bertindak sebagai gerbang yang memeriksa keberadaan token valid di penyimpanan lokal peramban (*LocalStorage*) sebelum mengizinkan pengguna mengakses halaman privat seperti *Dashboard* atau *Cart*. Jika token tidak ditemukan atau kadaluwarsa, pengguna secara otomatis diarahkan kembali ke halaman *Login*.



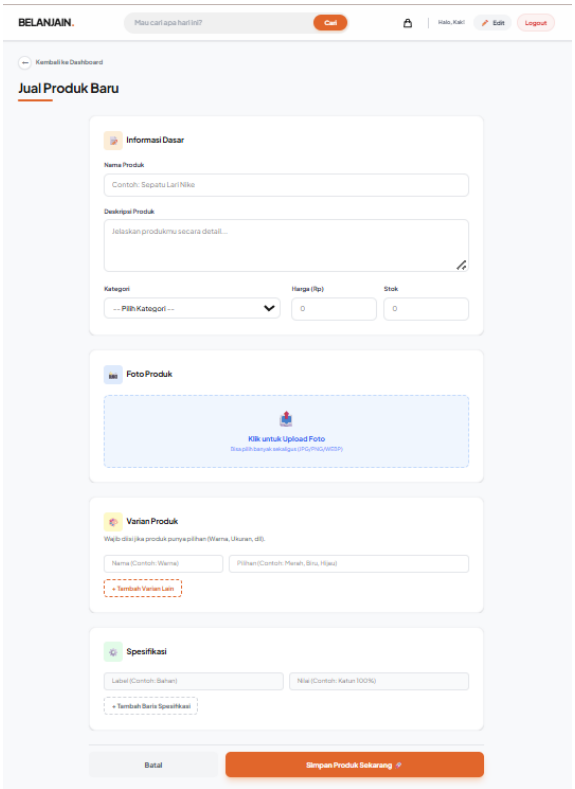
	id	email	password	name	role	storeName	storeLocation	hashedRefreshToken	createdAt	updatedAt
Edit Copy Delete	1	halozerly@gmail.com	$2b$10$O9muVr1qcnMxxBNvRjAC4Ozxf5B57QLTx81y24/rGsj...	Zerly	SELLER	Lo Clothes Official Store	Jalan Kanginan I, RW 01, Ketabang, Genteng, Suraba...	$2b$10$6jkE55UJzLXz4897jlFZ7.ocuZ8FDd.qTLZt191e1Hn...	2025-12-10 14:16:53.206	2025-12-10 14:32:30.569
Edit Copy Delete	2	sneakers@gmail.com	$2b$10$EmjA6IBBmMLrWFW9yMpl9uBowxWWpqOwTvkpWJyVrT2...	Sneakers Flash	SELLER	Sneakers Flash	Sepuluh Nopember Institute of Technology, Jalan Ke...	$2b$10$e1BGjbEcnWyFMrx34/Eh1O1Yaf2sZTJ34BgreXR6HF...	2025-12-10 14:27:54.565	2025-12-10 14:27:54.637
Edit Copy Delete	3	gramedia@gmail.com	$2b$10$/NI0Q867JCaJwmgfOev.7ujVYd6T9zSenWnECCmAwaL...	Gramedia	SELLER	Gramedia Official Shop	Sepuluh Nopember Institute of Technology, Kerfajay...	$2b$10$IWuQ/BJhOfSSPwTJXfFaTeFPbqJw4U7GI.CQYO4Rcyv...	2025-12-10 14:33:54.162	2025-12-10 14:33:54.223

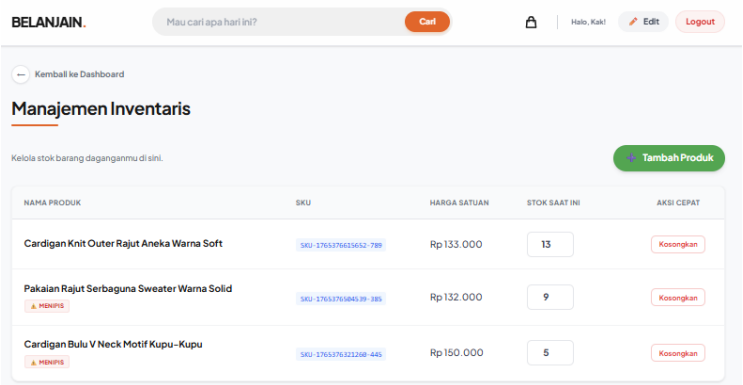
4.2 Manajemen Produk dan Inventaris

Fitur manajemen produk dirancang untuk memberikan fleksibilitas tinggi bagi penjual dalam mendeskripsikan barang dagangannya. Implementasi ini melibatkan penanganan data teks dan data biner (gambar) secara bersamaan.

Di sisi server, ProductsController memanfaatkan *Interceptor* dari pustaka Multer untuk menangani unggahan berkas gambar. Sistem memvalidasi tipe berkas (harus gambar) dan ukuran berkas sebelum menyimpannya ke penyimpanan lokal server dan mencatat jalur aksesnya (*file path*) ke basis data. Salah satu keunggulan teknis dalam modul ini adalah penggunaan tipe data JSON pada kolom variants dan specifications di tabel MySQL. Hal ini memungkinkan penjual mendefinisikan atribut produk yang dinamis—seperti warna, ukuran, atau spesifikasi teknis—tanpa memerlukan struktur tabel relasional yang kaku dan kompleks.

Untuk sisi antarmuka, halaman "Tambah Produk" menggunakan objek FormData untuk mengemas data teks dan fail gambar sebelum dikirim ke API. Selain itu, fitur unggulan "Manajemen Inventaris Cepat" (*Quick Inventory Management*) diimplementasikan pada halaman MyInventory.tsx. Halaman ini menyajikan tabel stok interaktif di mana penjual dapat mengubah angka stok secara langsung (*inline editing*) dan menyimpannya tanpa perlu berpindah halaman. Logika antarmuka juga memberikan umpan balik visual berupa indikator warna merah pada kolom input jika stok produk berada di level kritis (di bawah 10 unit) atau habis, memudahkan penjual untuk segera melakukan tindakan *restock*.





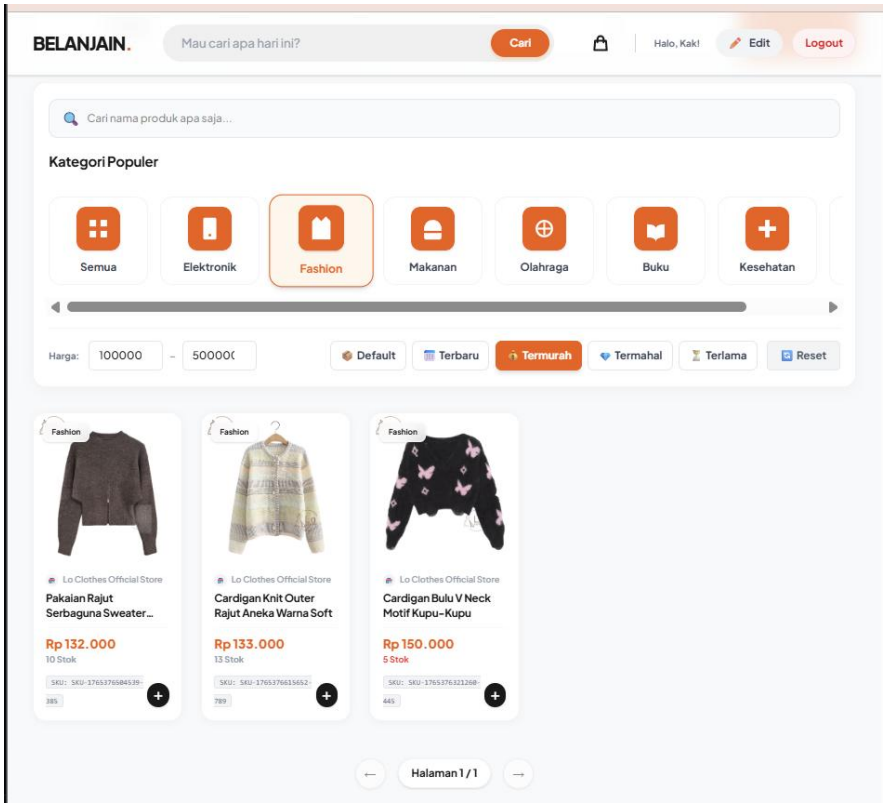
4.3 Alur Belanja dan Transaksi Kompleks (Split Order)

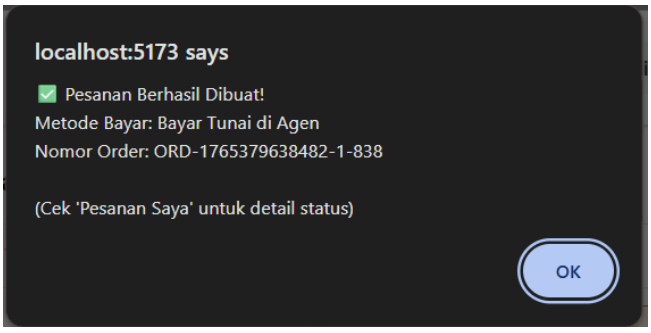
Alur belanja pelanggan merupakan bagian sistem dengan logika bisnis paling kompleks, terutama pada tahap pencarian dan penyelesaian transaksi (*Checkout*).

Fitur pencarian dan penyaringan (*Search & Filtering*) diimplementasikan menggunakan kueri basis data yang dinamis pada ProductsService. Sistem mampu menyusun kueri WHERE secara bersyarat berdasarkan parameter yang dikirim dari *Frontend*, seperti kata kunci nama produk, rentang harga minimum-maksimum, kategori, hingga pengurutan data (*Sorting*). Hal ini memberikan respons hasil pencarian yang akurat dan relevan bagi pembeli.

Puncak kompleksitas sistem terletak pada fitur Split Order saat proses *Checkout*. Dalam skenario *marketplace* nyata, seorang pembeli dapat memasukkan produk dari berbagai toko (penjual) yang berbeda ke dalam satu keranjang belanja. Sistem "Belanjain" menangani hal ini dengan algoritma pemecahan pesanan di *Backend*. Ketika tombol *Checkout* ditekan:

1. Sistem menerima daftar barang yang akan dibeli.
2. Sistem mengelompokkan barang-barang tersebut berdasarkan ID Penjual (sellerId).
3. Menggunakan Prisma Interactive Transaction (prisma.$transaction), sistem secara atomik membuat beberapa entitas Order yang terpisah untuk masing-masing penjual, meskipun berasal dari satu kali aksi klik pembeli.
4. Secara bersamaan, sistem mengurangi stok produk terkait dan menghapus barang dari keranjang belanja. Penggunaan transaksi basis data menjamin integritas data: jika salah satu proses gagal (misalnya stok habis di tengah proses), seluruh transaksi akan dibatalkan (*rollback*), mencegah terjadinya ketidakkonsistenan data keuangan atau stok.



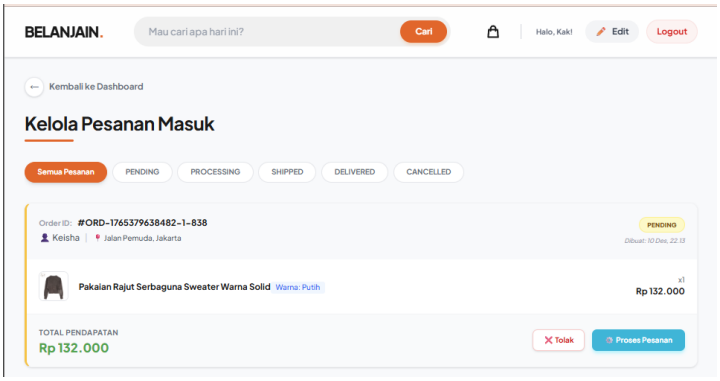


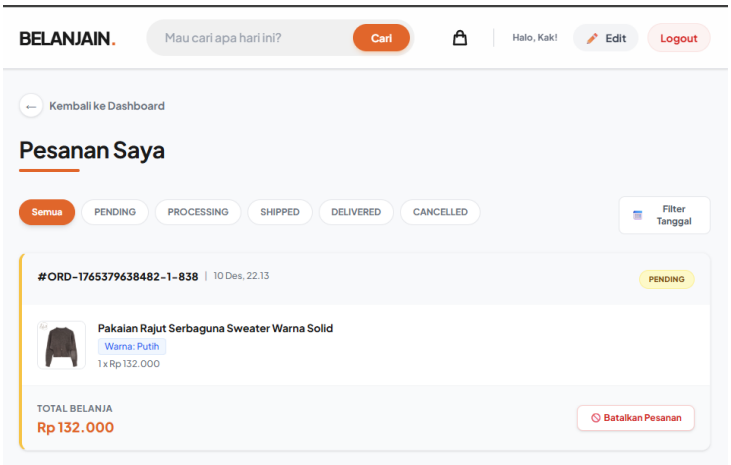
4.4 Manajemen Siklus Hidup Pesanan

Setelah pesanan terbentuk, sistem menyediakan alur kerja (*workflow*) manajemen status yang terstruktur. Siklus hidup pesanan bergerak dari status PENDING (Menunggu

diproses), PROCESSING (Sedang disiapkan), SHIPPED (Dikirim), hingga DELIVERED (Diterima) atau CANCELLED (Dibatalkan).

Bagi penjual, halaman ManageOrders menyediakan kontrol penuh untuk memajukan status pesanan. Logika di *Backend* memvalidasi setiap transisi status, misalnya memastikan bahwa pesanan yang sudah dikirim tidak bisa dibatalkan secara sepihak oleh penjual. Bagi pembeli, halaman MyOrders memberikan transparansi status terkini. Fitur penting yang diimplementasikan di sini adalah tombol "Konfirmasi Terima Barang" yang hanya muncul ketika status pesanan adalah SHIPPED. Ketika tombol ini ditekan, status berubah menjadi DELIVERED dan sistem mencatat tanggal penyelesaian transaksi, yang selanjutnya akan digunakan dalam perhitungan pendapatan penjual. Selain itu, jika pembeli membatalkan pesanan yang masih PENDING, sistem secara otomatis mengembalikan jumlah stok barang ke inventaris toko (*Auto-Restock*).

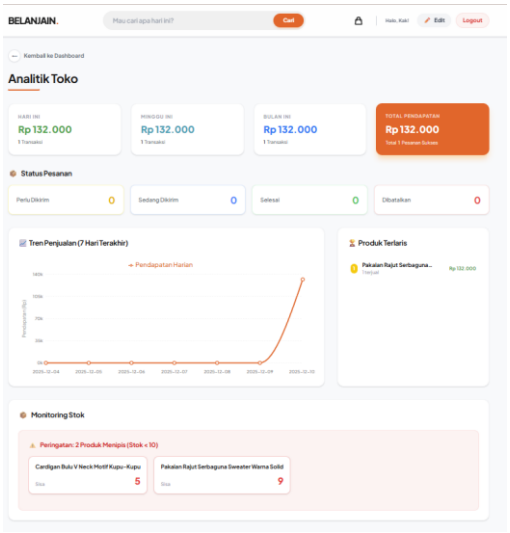


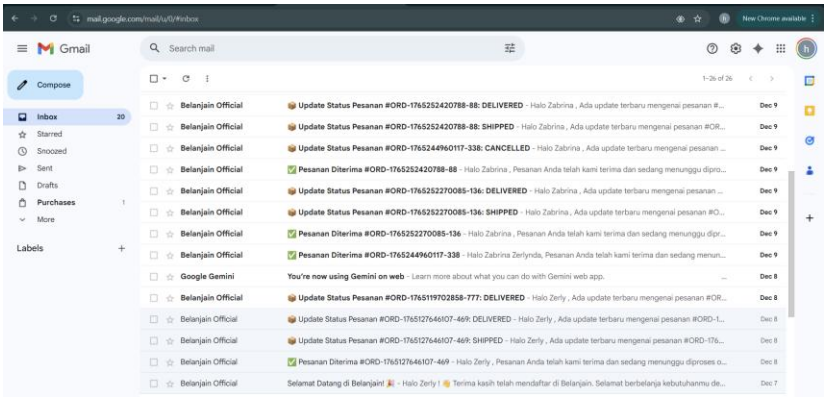


4.5 Dashboard Analitik dan Otomatisasi Sistem

Halaman Seller Analytics menyajikan visualisasi data yang mendalam. Data mentah transaksi diolah menggunakan fungsi agregasi basis data (groupBy, sum, count) di *Backend* untuk menghasilkan metrik kunci seperti Total Pendapatan Harian, Mingguan, dan Bulanan. Data tren penjualan selama 7 hari terakhir divisualisasikan menggunakan pustaka grafik Recharts di *Frontend*, memberikan representasi visual yang mudah dipahami mengenai naik-turunnya performa toko. Selain itu, sistem juga menampilkan daftar produk terlaris (*Top Selling Products*) berdasarkan akumulasi kuantitas terjual.

Di balik layar, sistem menjalankan fitur otomatisasi menggunakan Cron Jobs melalui modul @nestjs/schedule. Sebuah layanan latar belakang (TasksService) dijadwalkan untuk berjalan secara otomatis pada waktu tertentu (misalnya setiap hari Senin pukul 09:00 pagi). Layanan ini akan memindai seluruh transaksi yang terjadi selama satu minggu terakhir, menghitung total pendapatan per penjual, dan mengirimkan Laporan Penjualan Mingguan (*Weekly Sales Report*) langsung ke alamat email penjual menggunakan layanan SMTP (Nodemailer). Fitur ini membuktikan bahwa aplikasi tidak hanya pasif menunggu input pengguna, tetapi juga mampu bekerja secara proaktif dan otomatis.





BAB V
PENGUJIAN DAN HASIL

Tahapan pengujian merupakan fase kritis dalam siklus pengembangan perangkat lunak yang bertujuan untuk menjamin bahwa seluruh komponen sistem "Belanjain" berfungsi sesuai dengan spesifikasi kebutuhan yang telah dirancang. Pengujian pada proyek ini dilakukan secara komprehensif menggunakan pendekatan *Black Box Testing* melalui metode *End-to-End (E2E) Testing* otomatis, serta analisis *White Box Testing* melalui pengukuran cakupan kode (*Code Coverage*). Metode E2E dipilih untuk mensimulasikan perilaku pengguna nyata dari awal hingga akhir proses pada level API, memastikan integritas data antar-modul, serta memvalidasi logika bisnis yang kompleks seperti pembagian pesanan (*Split Order*) dan kalkulasi analitik. Pengujian dieksekusi menggunakan kerangka kerja Jest dan Supertest yang berjalan di lingkungan *backend* NestJS, memastikan setiap baris kode yang kritikal telah diverifikasi kebenarannya.

5.1 Pengujian Modul Autentikasi dan Manajemen Pengguna

Skenario pengujian pertama difokuskan pada keamanan sistem dan manajemen sesi pengguna, sebagaimana tercatat dalam berkas pengujian products.e2e-spec.ts (bagian Auth) dan users.e2e-spec.ts. Pengujian mencakup validasi alur pendaftaran (*Register*) dan masuk (*Login*), di mana sistem berhasil menerbitkan Token JWT saat kredensial valid dan menolak akses dengan kode status 401 Unauthorized atau 403 Forbidden saat terjadi kesalahan input atau pelanggaran hak akses. Fitur keamanan lanjutan, yaitu mekanisme *Refresh Token*, juga terbukti berhasil memperpanjang sesi pengguna tanpa mengharuskan *login* ulang. Selain itu, pengujian pada modul profil pengguna memastikan bahwa pengguna dapat memperbarui informasi pribadi, termasuk nama toko dan kata sandi, dengan respons waktu yang sangat cepat (rata-rata di bawah 200 milidetik). Keberhasilan seluruh 26 kasus uji (*test cases*) pada sektor ini mengonfirmasi bahwa fondasi keamanan aplikasi telah terbangun dengan kokoh.

5.2 Pengujian Manajemen Produk, Konten, dan Inventaris

Pengujian pada sektor manajemen produk melibatkan skenario yang komprehensif, mencakup siklus hidup konten dari penciptaan hingga penghapusan, yang terekam dalam content-management.e2e-spec.ts dan inventory.e2e-spec.ts. Sistem diuji kemampuannya dalam menangani operasi CRUD (*Create, Read, Update, Delete*) produk, termasuk validasi tipe data dan unggahan gambar. Poin krusial yang berhasil divalidasi adalah logika kepemilikan data, di mana sistem secara efektif mencegah pengguna mengubah atau menghapus produk yang bukan miliknya. Lebih lanjut, fitur manajemen inventaris (*Inventory Management*) diuji secara spesifik untuk memverifikasi logika bisnis *restock*, penandaan stok habis (*Out of Stock*), dan pemicu peringatan stok menipis (*Low Stock Alerts*). Hasil pengujian menunjukkan bahwa sistem mampu mendeteksi ambang batas stok (kurang dari 10 unit) dan memberikan respons yang sesuai, serta menolak input jumlah stok yang tidak valid (negatif).

5.3 Pengujian Transaksi dan Logika Split Order

Bagian ini merupakan inti dari kompleksitas sistem "Belanjain", di mana pengujian dilakukan terhadap modul Keranjang (cart.e2e-spec.ts) dan Pesanan (orders.e2e-spec.ts). Pada modul keranjang, sistem berhasil menangani penambahan produk dengan varian yang berbeda (misalnya warna atau ukuran) sebagai entitas item yang terpisah, serta melakukan kalkulasi total harga secara akurat. Pengujian pada modul pesanan membuktikan keberhasilan implementasi fitur *Split Order*. Dalam skenario uji coba di mana satu keranjang berisi produk dari berbagai penjual, sistem terbukti mampu memecah transaksi tersebut menjadi beberapa Nomor Order yang unik secara otomatis dalam satu proses *checkout*. Selain itu, siklus hidup pesanan mulai dari PENDING, pembatalan pesanan yang mengembalikan stok otomatis (*Restock on Cancel*), hingga penyelesaian pesanan (DELIVERED) yang memicu pengiriman email notifikasi, seluruhnya berjalan sukses tanpa kegagalan (*Passed*), menjamin integritas data transaksi multi-penjual.

5.4 Pengujian Dasbor Analitik dan Ulasan

Untuk memvalidasi fitur pendukung keputusan bagi penjual, pengujian dilakukan terhadap modul Dasbor (dashboard.e2e-spec.ts) dan Ulasan (reviews.e2e-spec.ts). Hasil pengujian mengonfirmasi akurasi agregasi data penjualan, di mana sistem berhasil menghitung total pendapatan harian, mingguan, dan bulanan, serta menyajikan data tren penjualan dan produk terlaris (*Top Selling Products*) dengan tepat. Fitur otomatisasi laporan mingguan (*Weekly Sales Report*) juga terbukti dapat dipicu sesuai jadwal. Di sisi interaksi sosial, sistem ulasan produk berhasil membatasi akses pemberian ulasan hanya kepada pembeli yang telah menyelesaikan transaksi (*Verified Purchase*), serta mencegah duplikasi ulasan untuk produk yang sama. Keberhasilan pengujian ini membuktikan bahwa sistem tidak hanya andal dalam transaksi, tetapi juga mampu menyajikan data analitik yang valid dan dapat dipercaya.

5.5 Rekapitulasi Hasil Pengujian dan Analisis Cakupan Kode

Berdasarkan eksekusi keseluruhan *test suite*, sistem "Belanjain" berhasil melewati total 63 skenario pengujian (Test Cases) yang tersebar di 8 berkas pengujian utama dengan total waktu eksekusi sekitar 14.7 detik. Seluruh pengujian menunjukkan status PASS, menandakan tidak adanya kegagalan logika (*logic failure*) maupun kesalahan sistem (*runtime error*).

Selain verifikasi fungsional, analisis kualitas kode juga dilakukan melalui pengukuran Code Coverage. Hasil analisis menunjukkan tingkat cakupan kode rata-rata (*All files*) mencapai 86.9% Statement Coverage, yang mengindikasikan bahwa sebagian besar instruksi dalam kode program telah dieksekusi selama pengujian. Beberapa modul inti menunjukkan tingkat cakupan yang sangat impresif, antara lain modul Dashboard dengan cakupan 99.13%, modul Users sebesar 96.66%, dan modul Reviews sebesar 96.15%. Tingginya angka cakupan pada modul *Dashboard* dan *Users* memberikan jaminan bahwa logika perhitungan analitik dan manajemen data pengguna—yang merupakan fitur vital aplikasi—telah teruji secara menyeluruh dan minim dari risiko *bug* tersembunyi.

```
PASS  test/products.e2e-spec.ts (6.754 s)
PASS  test/orders.e2e-spec.ts
PASS  test/inventory.e2e-spec.ts
PASS  test/cart.e2e-spec.ts
PASS  test/content-management.e2e-spec.ts
PASS  test/users.e2e-spec.ts
PASS  test/dashboard.e2e-spec.ts
PASS  test/reviews.e2e-spec.ts
```

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	86.9	71.46	87.71	88.61	
src	84.61	75	33.33	77.77	
app.controller.ts	87.5	75	50	83.33	10
app.service.ts	80	100	0	66.66	6
src/auth	90.09	76	90.47	91.76	
auth.controller.ts	92.3	75	85.71	91.66	34-35
auth.service.ts	85.41	76.92	87.5	90.24	85-96
jwt.strategy.ts	100	100	100	100	
roles.decorator.ts	100	100	100	100	
roles.guard.ts	93.33	75	100	91.66	15
src/cart	89.06	72.91	100	91.22	
cart.controller.ts	100	75	100	100	10-26
cart.service.ts	85.1	72.22	100	88.09	30,61,122,129,147
src/dashboard	99.13	78.57	93.75	99.03	
dashboard.controller.ts	100	75	100	100	8
dashboard.service.ts	98.73	82.14	90	98.64	108
tasks.service.ts	100	70	100	100	11-53
src/email	13.88	0	0	8.82	
email.service.ts	13.88	0	0	8.82	9-195
src/orders	86.92	63.38	100	94.59	
orders.controller.ts	100	75	100	100	10-32
orders.service.ts	84.4	61.01	100	93.47	50,220-221,232,234-235
src/prisma	100	100	100	100	
prisma.service.ts	100	100	100	100	
src/products	88.28	79.06	100	90.17	
products.controller.ts	87.71	78.12	100	87.03	41,69-73,83-87
products.service.ts	88.73	79.62	100	93.1	66,119,128,155
src/reviews	96.15	75	100	95.45	
reviews.controller.ts	100	75	100	100	8-13
reviews.service.ts	92.85	75	100	91.66	23
src/users	96.66	77.77	100	100	
users.controller.ts	100	75	100	100	9-19
users.service.ts	94.44	80	100	100	8-23

```
Test Suites: 8 passed, 8 total
Tests:       63 passed, 63 total
Snapshots:   0 total
Time:        14.745 s, estimated 20 s
Ran all test suites.
```

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan seluruh tahapan yang telah dilalui, mulai dari analisis kebutuhan, perancangan sistem, implementasi kode, hingga pengujian komprehensif, dapat ditarik beberapa kesimpulan utama mengenai pengembangan sistem E-commerce "Belanjain":

1. Keberhasilan Implementasi Arsitektur Full-Stack: Sistem berhasil dibangun menggunakan arsitektur modern yang memisahkan *Frontend* (React) dan *Backend* (NestJS). Penerapan pola ini terbukti meningkatkan modularitas kode, mempermudah proses *debugging*, dan menghasilkan performa aplikasi yang responsif melalui pendekatan *Single Page Application* (SPA).
2. Pemenuhan Kebutuhan Fungsional dan Fitur Lanjutan: Seluruh kebutuhan fungsional yang ditargetkan, mencakup 10 *User Stories* utama dan fitur tambahan (Profil Pengguna & Halaman Toko), telah berhasil diimplementasikan. Fitur unggulan Split Order terbukti mampu menangani logika transaksi kompleks multi-penjual secara atomik menggunakan transaksi basis data, menjaga integritas data keuangan dan stok tanpa kesalahan.
3. Efektivitas Otomatisasi dan Analitik: Integrasi fitur otomatisasi menggunakan *Cron Jobs* dan layanan email (SMTP) berjalan efektif dalam mengurangi beban kerja manual penjual. Sistem mampu mengirimkan notifikasi *real-time* dan laporan penjualan mingguan secara mandiri. Selain itu, Dasbor Analitik berhasil menyajikan visualisasi data yang akurat untuk mendukung pengambilan keputusan bisnis.
4. Keandalan dan Kualitas Kode: Hasil pengujian otomatis (*End-to-End Testing*) menunjukkan tingkat keberhasilan 100% pada 63 skenario uji tanpa adanya kegagalan. Didukung dengan tingkat cakupan kode (*Code Coverage*) mencapai rata-rata 86.9% (bahkan mencapai >99% pada modul inti seperti Dashboard), sistem ini dapat dinyatakan memiliki kualitas kode yang tinggi, minim *bug*, dan aman untuk digunakan.

6.2 Saran

Meskipun sistem "Belanjain" telah memenuhi seluruh spesifikasi yang ditetapkan, terdapat beberapa peluang pengembangan untuk meningkatkan kapabilitas sistem di masa mendatang:

1. Integrasi Pembayaran Digital (*Payment Gateway*): Saat ini sistem menggunakan simulasi pembayaran. Pengembangan selanjutnya disarankan mengintegrasikan layanan *Payment Gateway* nyata (seperti Midtrans atau Xendit) untuk memfasilitasi pembayaran otomatis melalui Virtual Account, E-Wallet, atau Kartu Kredit secara *real-time*.
2. Kalkulasi Logistik Terintegrasi: Mengimplementasikan API pihak ketiga (seperti RajaOngkir) untuk menghitung biaya pengiriman secara dinamis berdasarkan berat produk, dimensi, dan lokasi riil penjual-pembeli, menggantikan sistem ongkos kirim statis.

3. Fitur Komunikasi Waktu Nyata (*Real-time Chat*): Menambahkan fitur percakapan langsung (*Live Chat*) antara penjual dan pembeli menggunakan teknologi WebSocket (Socket.io). Hal ini akan mempercepat proses negosiasi atau tanya-jawab produk tanpa perlu berpindah ke aplikasi pesan eksternal.
4. Pengembangan Aplikasi Seluler (*Mobile App*): Mengingat arsitektur *Backend* sudah berbasis REST API, pengembangan aplikasi *mobile* (Android/iOS) menggunakan React Native atau Flutter dapat dilakukan dengan mudah. Hal ini akan memungkinkan fitur notifikasi dorong (*Push Notification*) yang lebih interaktif dibandingkan email.
5. Penyebaran ke Cloud (*Deployment*): Untuk kebutuhan produksi, sistem disarankan untuk di-*deploy* menggunakan layanan Cloud (seperti AWS, Google Cloud, atau Vercel) dengan penerapan kontainerisasi (Docker) dan orkestrasi CI/CD untuk menjamin ketersediaan layanan (*high availability*) dan kemudahan pembaruan sistem.