

BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Penelitian ini didasarkan pada berbagai penelitian terdahulu yang membahas manajemen rantai pasok (*Supply Chain Management*) dan integrasi sistem pembayaran digital pada ekosistem bisnis waralaba.

Penelitian yang dilakukan oleh Melisa, dkk [22]. berfokus pada pengembangan sistem inventaris gudang berbasis *web* dengan menerapkan metode *Waterfall* untuk mengatasi permasalahan pengelolaan stok manual. Studi ini berhasil merancang sistem yang mampu mencatat barang masuk dan keluar secara terstruktur, meningkatkan akurasi data stok, serta memfasilitasi pelaporan inventaris yang lebih cepat dan efisien dibandingkan metode konvensional. Sistem ini memberikan kontribusi signifikan dalam meminimalisir kesalahan pencatatan dan memudahkan pemantauan aset perusahaan secara digital. Namun, fokus utama penelitian ini masih terbatas pada aspek pencatatan logistik semata dan belum terintegrasi dengan fitur pembayaran digital untuk *restock* barang, sehingga proses penyelesaian transaksi pengadaan barang masih harus dilakukan melalui saluran terpisah.

Selanjutnya dalam ranah pencatatan transaksi penjualan, Pratama, Dermawan, dan Carudin [23] mengembangkan aplikasi *Point of Sales* (POS) berbasis *website* yang dilengkapi dengan fitur manajemen katalog produk dan laporan penjualan otomatis. Penelitian ini menawarkan solusi

digitalisasi pencatatan transaksi yang terbukti meningkatkan efisiensi operasional toko dan akurasi data keuangan. Keunggulan sistem ini terletak pada kemampuannya mengelola transaksi penjualan secara *real-time* dan modern. Akan tetapi, keterbatasan sistem ini adalah hanya memantau penjualan harian ke konsumen akhir, dan belum mengatur suplai bahan baku (*Supply Chain Management*) dari pusat, sehingga belum mampu mengakomodasi kebutuhan bisnis dengan struktur rantai pasok yang kompleks seperti waralaba.

Terkait implementasi pembayaran digital, Chan, dkk [24]. merancang sebuah *website e-commerce* dengan mengintegrasikan *Payment Gateway Midtrans*. Penelitian ini berhasil memfasilitasi pemesanan barang dan jasa secara daring dengan verifikasi pembayaran otomatis, yang secara signifikan mengurangi risiko kesalahan *input* manual dan mempercepat proses transaksi pelanggan. Integrasi *Midtrans* dalam sistem ini memberikan keamanan dan kenyamanan lebih melalui berbagai opsi pembayaran non-tunai. Meskipun demikian, implementasi ini memiliki kekurangan karena fokusnya hanya transaksi B2C (ke pelanggan), bukan B2B (antar mitra *franchise*), sehingga belum mencakup mekanisme pembayaran tagihan atau royalti antar entitas bisnis yang saling bermitra.

Dalam konteks sistem yang lebih kompleks, Hadikin dan Wiratama[25] mengembangkan prototipe sistem *Enterprise Resource Planning* (ERP) berbasis *web* yang mengintegrasikan modul inventaris dan *Supply Chain Management* (SCM) untuk industri logistik. Penelitian ini

menonjolkan kemampuan sistem dalam merampingkan proses manual, mengurangi kesalahan manusia, dan meningkatkan efisiensi rantai pasok melalui pemantauan pergerakan barang yang terintegrasi. Sistem ini menawarkan visibilitas yang baik terhadap aliran barang dari gudang hingga distribusi. Namun, kelemahan dari penelitian ini adalah belum ada yang menggabungkan manajemen stok (SCM) dan *Payment Gateway* dalam satu platform terpadu, di mana aspek penyelesaian finansial otomatis untuk setiap pergerakan stok antar entitas belum diakomodasi dalam sistem tersebut.

Berdasarkan tinjauan terhadap penelitian-penelitian terdahulu, terlihat bahwa pengembangan sistem saat ini cenderung berjalan secara parsial; beberapa berfokus pada manajemen stok gudang, beberapa pada pemantauan penjualan (POS), dan lainnya pada integrasi pembayaran digital ke pelanggan akhir (B2B). Belum ditemukan penelitian yang secara spesifik membangun sistem yang menyatukan manajemen rantai pasok (SCM) untuk distribusi stok dari pusat ke cabang dengan fitur pembayaran digital otomatis (B2B). Oleh karena itu, penelitian ini hadir untuk menutupi kekurangan penelitian-penelitian di atas dengan menggabungkan *SCM* dan *Midtrans* secara simultan, guna menciptakan ekosistem bisnis waralaba yang efisien, transparan, dan terintegrasi baik dari sisi logistik maupun finansial.

2.2 Landasan Teori

2.2.1 Supply Chain Management (SCM)

Sistem *Supply Chain Management* (SCM) didefinisikan sebagai suatu pendekatan strategis yang bertujuan untuk mengintegrasikan berbagai elemen organisasi termasuk pemasok (supplier), manufaktur, distributor, ritel, hingga pelanggan agar tercipta proses bisnis yang terstruktur dan efisien. Fokus utama *SCM* adalah pencatatan dan pengelolaan data barang, dimulai dari pengadaan bahan baku, proses produksi, hingga pengiriman barang ke pelanggan akhir. Tujuan penerapan *SCM* adalah untuk mencapai efisiensi biaya sistem secara keseluruhan seminimal mungkin sekaligus memenuhi tingkat layanan (service level) yang diinginkan. Dalam konteks operasional sistem informasi, penerapan *SCM* berfungsi mengintegrasikan seluruh bagian operasional untuk meminimalisir kesalahan pemesanan (human error) dan mengotomatisasi informasi persediaan agar pasokan barang antar mitra berjalan terkendali [26].

2.2.2 Sistem Informasi Berbasis Web

Sistem Informasi berbasis *web* merupakan teknologi informasi yang memungkinkan pengelolaan proses bisnis dilakukan secara terstruktur dan efisien melalui jaringan internet. Sistem ini memudahkan pengguna, baik admin pusat maupun mitra cabang, untuk mengakses data dan laporan secara *real-time* dari berbagai lokasi tanpa batasan geografis [27].

2.2.3 Framework Laravel

Laravel merupakan *framework* pengembangan aplikasi berbasis bahasa pemrograman *PHP* yang dirancang untuk mengoptimalkan proses pembuatan *website*. *Laravel* memiliki filosofi yang berfokus pada kejelasan dan kesederhanaan (*clarity and simplicity*), baik dalam penulisan kode maupun tampilan antarmuka. Keunggulan utama *Laravel* meliputi struktur kode yang rapi (MVC), fitur keamanan yang kuat, serta dukungan *Migrations* dan *Seeder* yang mempermudah pengelolaan basis data dalam sistem yang kompleks [28].

2.2.4 Tailwind CSS

Tailwind CSS merupakan pustaka atau kerangka kerja (*framework*) berbasis *Cascading Style Sheets* (CSS) yang mengusung pendekatan *utility-first* untuk mempermudah pengembang dalam membangun antarmuka pengguna (user interface) langsung melalui penyematan kelas-kelas utilitas pada elemen *HTML*. Konsep *utility-first* ini bekerja dengan menyediakan sekumpulan kelas siap pakai yang terstruktur, sehingga memungkinkan perancangan tata letak (layout) dan modifikasi gaya visual *website* dapat dilakukan secara cepat tanpa keharusan menulis gaya CSS tradisional dari awal. Keunggulan utama dari penggunaan *Tailwind CSS* terletak pada kemampuannya untuk mempercepat proses implementasi desain secara signifikan, sekaligus memberikan kebebasan dan fleksibilitas

tinggi dalam menyesuaikan tampilan agar menjadi responsif, sehingga antarmuka sistem dapat berjalan optimal dan nyaman diakses melalui berbagai ukuran layar perangkat [29].

2.2.5 Basis Data MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data relasional atau *Relational Database Management System* (RDBMS) yang bersifat *open-source* dan dirancang khusus untuk memfasilitasi pengoperasian serta pengelolaan data secara terstruktur. Dalam ekosistem pengembangan sistem informasi berbasis *web*, MySQL memiliki fungsi krusial sebagai wadah penyimpanan kumpulan data secara sistematis agar informasi tersebut dapat dengan mudah diakses, dimanipulasi, maupun diperbarui sesuai kebutuhan operasional aplikasi. Penggunaan RDBMS ini sangat penting karena MySQL mampu mendukung pengelolaan relasi data yang kompleks secara konsisten, menjamin tingkat keamanan data yang tinggi, serta dapat terintegrasi langsung dengan bahasa pemrograman *server-side* (seperti PHP) untuk mendukung terciptanya aplikasi *web* yang dinamis dan interaktif [30].

2.2.6 Payment Gateway (Midtrans)

Payment Gateway merupakan teknologi jembatan yang menghubungkan situs *web* pedagang (*merchant*) dengan lembaga keuangan untuk memproses dan mengotorisasi transaksi pembayaran daring (*online*). Fungsi utamanya adalah melegalkan informasi

transaksi, mengamankan data pembayaran melalui enkripsi, serta mendeteksi potensi penipuan (fraud detection). Dalam penelitian ini, *Payment Gateway* yang digunakan adalah *Midtrans*. *Midtrans* merupakan penyedia layanan gerbang pembayaran yang memfasilitasi berbagai metode transaksi seperti transfer bank (Virtual Account), kartu kredit, hingga dompet digital (E-Wallet). *Midtrans* menyediakan fitur lingkungan pengujian yang disebut *Sandbox*. Fitur ini memungkinkan pengembang melakukan simulasi transaksi pembayaran menggunakan data tiruan (dummy) untuk memastikan fungsionalitas sistem berjalan sesuai spesifikasi sebelum diterapkan pada transaksi riil [31].

2.2.7 Unified Modeling Language (UML)

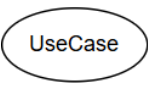
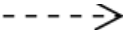
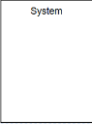
Unified Modeling Language (UML) merupakan standar bahasa pemodelan yang digunakan untuk merancang dan mendokumentasikan sistem perangkat lunak berorientasi objek. Diagram *UML* yang digunakan dalam penelitian ini meliputi [32]:

1. Use Case Diagram

Use Case Diagram mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang dibuat. Diagram ini digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang

digunakan dalam *Use Case Diagram* dapat dilihat pada Tabel 2.1.

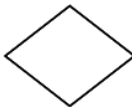
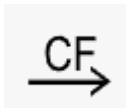
Tabel 2. 1 Simbol Use Case Diagram.

No.	Simbol	Nama	Keterangan
1		<i>Actor</i>	Mewakili peran orang, sistem lain, atau alat yang berinteraksi dengan sistem.
2		<i>Use Case</i>	Fungsionalitas atau layanan yang disediakan oleh sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.
3		<i>Association</i>	Menghubungkan <i>link</i> antar elemen, biasanya antara aktor dan <i>use case</i> .
4		<i>Include</i>	Menunjukkan bahwa suatu <i>use case</i> seluruhnya merupakan fungsionalitas dari <i>use case</i> lainnya.
5		<i>System</i>	Mendefinisikan paket untuk menggambarkan sistem secara terbatas.

2. Activity Diagram

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Diagram ini menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Simbol-simbol yang digunakan dalam *Activity Diagram* dapat dilihat pada Tabel 2.2.

Tabel 2. 2 Simbol Activity Diagram

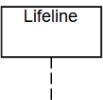
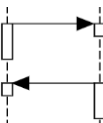
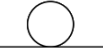
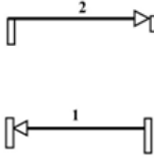
No	Simbol	Nama	Keterangan
1		<i>Start Point</i>	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2		<i>End Point</i>	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
3		<i>Activity</i>	Aktivitas yang dilakukan sistem, biasanya diawali dengan kata kerja.
4		<i>Decision</i>	Asosiasi percabangan di mana jika ada pilihan aktivitas lebih dari satu.
5		<i>Control Flow</i>	Arah aliran aktivitas dari satu status ke status lainnya.

3. Sequence Diagram

Sequence Diagram menggambarkan interaksi antar objek di *dalam* dan di sekitar sistem (termasuk pengguna dan *display*) berupa pesan (*message*) yang digambarkan terhadap waktu. Diagram ini digunakan untuk memodelkan skenario atau langkah-langkah yang dilakukan secara berurutan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu, seperti pada proses validasi pembayaran otomatis. Simbol-

simbol yang digunakan dalam *Sequence Diagram* dapat dilihat pada Tabel 2.3.

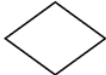
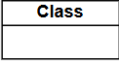
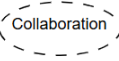
Tabel 2. 3 Simbol Sequence Diagram

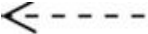
No	Simbol	Nama	Keterangan
1.		<i>Lifeline</i>	Objek entity, antara muka yang saling berinteraksi.
2.		<i>Message</i>	Spesifikasi dari komunikasi antar objek yang memuat informasi tentang aktivitas yang terjadi.
3.		<i>Actor</i>	Menggambarkan orang yang sedang berinteraksi dengan <i>system</i> .
4.		<i>Boundary Class</i>	Menggambarkan penggambaran dari form.
5.		<i>Entity Class</i>	Menggambarkan hubungan kegiatan yang akan dilakukan.
6.		<i>Control Class</i>	Menggambarkan penghubung antara Boundary dengan label.
7.		<i>Activation</i>	Sebagai sebuah objek yang akan melakukan sebuah aksi.
8.		<i>Message</i>	Mengindikasikan komunikasi antar objek dengan objek.
9.		<i>Self Message</i>	Mengindikasikan komunikasi kembali kedalam sebuah objek itu sendiri.

4. Class Diagram

Class Diagram merupakan diagram struktural yang memberikan gambaran tentang struktur sistem, deskripsi kelas (*class*), *package*, serta hubungan (relasi) antar objek seperti pewarisan, asosiasi, dan lain-lain. Dalam pengembangan aplikasi berbasis *web*, *Class Diagram* sering digunakan sebagai representasi logis dari struktur basis data dan relasi antar tabel. Simbol-simbol yang digunakan dalam *Class Diagram* dapat dilihat pada Tabel 2.4.

Tabel 2. 4 Simbol Class Diagram

No	Simbol	Nama	Keterangan
1.		<i>Generalization</i>	Hubungan di mana objek anak (<i>descendant</i>) mewarisi perilaku dan struktur dari objek induk (<i>ancestor</i>).
2.		<i>Nary Association</i>	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3.		<i>Class</i>	Himpunan dari objek-objek yang berbagai atribut serta operasi yang sama.
4.		<i>Collaboration</i>	Deskripsi dari urutan aksi yang ditampilkan dalam sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
5.		<i>Dependency</i>	Operasi yang benar-benar dilakukan oleh suatu objek.

No	Simbol	Nama	Keterangan
6.		<i>Dependency</i>	Hubungan Dimana perubahan yang terjadi pada suatu elemen (<i>independent</i>) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
7.		<i>Association</i>	Apa yang menghubungkan antara objek satu dengan objek lainnya.

2.2.8 Metode Pengembangan Sistem (*Waterfall*)

Metode *pengembangan* sistem yang diterapkan merupakan *Software Development Life Cycle* (SDLC) dengan model *Waterfall*. Model ini memiliki karakteristik pengerjaan yang sistematis, berurutan, dan terukur. Tahapan dalam metode *Waterfall* meliputi [33]:

- a. *Requirement*: Analisis kebutuhan sistem melalui observasi dan wawancara.
- b. *Design*: Perancangan desain sistem, basis data, dan antarmuka.
- c. *Development*: Penulisan kode program (*coding*) sesuai desain.
- d. *Testing*: Pengujian sistem untuk memastikan tidak ada kesalahan (*bug*).
- e. *Maintenance*: Pemeliharaan dan perbaikan sistem pasca implementasi

2.2.9 Pengujian Sistem (Black Box Testing)

Pengujian sistem dilakukan menggunakan metode *Black Box Testing*. Metode ini berfokus pada pengujian fungsionalitas aplikasi tanpa melihat struktur kode internalnya. Pengujian dilakukan dengan cara memberikan berbagai skenario *input* pada fitur-fitur sistem untuk memastikan bahwa *output* yang dihasilkan sudah sesuai dengan spesifikasi kebutuhan dan bebas dari kesalahan logika [34].