# BAB II
# KAJIAN LITERATUR

## 2.1 Sistem Informasi

Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan dan berkumpul bersama untuk mencapai sasaran atau tujuan tertentu [5]. Secara lebih mendalam, sistem juga dapat didefinisikan sebagai serangkaian elemen atau komponen fisik maupun konseptual yang saling bergantung dan terintegrasi satu sama lain guna melaksanakan suatu kegiatan pokok di dalam sebuah entitas perusahaan [6]. Tanpa adanya sistem yang mengatur alur tersebut, setiap kegiatan operasional rentan kehilangan arah dan fungsi pengawasan.

Informasi merupakan hasil pengolahan data mentah melalui cara tertentu sehingga berubah menjadi bentuk yang memiliki makna dan kegunaan yang lebih baik bagi pihak penerimanya [7]. Proses transformasi data menjadi sebuah informasi yang akurat sangatlah krusial, karena hasil dari informasi tersebut bertindak sebagai pijakan utama bagi pihak manajemen perusahaan dalam memperbaiki proses perumusan kebijakan dan pengambilan keputusan harian [8].

Sistem informasi pada dasarnya diartikan sebagai sekumpulan prosedur formal yang menggabungkan komponen manusia, perangkat lunak, perangkat keras, dan data untuk mengumpulkan, memproses, serta menyebarkan informasi di dalam sebuah organisasi [9]. Integrasi operasional ini berfokus pada penyelesaian berbagai persoalan manajerial dengan memanfaatkan teknologi komputasi agar aliran rekam jejak pekerjaan menjadi jauh lebih transparan, aman, dan efisien [10].

Lebih jauh lagi, kehadiran sistem informasi secara krusial bertujuan untuk mendigitalisasi alur tata kerja yang sebelumnya dilakukan secara *manual* menggunakan kertas, sehingga perusahaan mampu menekan tingkat redundansi atau penggandaan pencatatan [11]. Penerapan ekosistem teknologi yang terstruktur dengan baik ini memungkinkan setiap unit divisi untuk mengakses laporan dan pembaruan riwayat data secara waktu nyata (*real-time*), yang pada akhirnya dapat mendongkrak mutu pelayanan dan efektivitas kerja secara maksimal [12].

## 2.2 Sistem Informasi Penjualan

Sistem informasi penjualan merupakan suatu perangkat lunak yang dirancang secara khusus untuk mengelola, merekam, dan memonitor seluruh sirkulasi perdagangan serta transaksi logistik pada suatu instansi bisnis [13]. Di dalam struktur sistem ini, fungsi pencatatan memegang peranan yang paling vital karena bertugas mendokumentasikan setiap nota pesanan, rincian suku cadang (*sparepart*), dan identitas pelanggan ke dalam satu basis data terpusat, sehingga risiko kehilangan dokumen fisik serta tingkat kealpaan input petugas administrasi (*human error*) dapat dicegah secara total [14].

Selain mengamankan data transaksi di titik awal, ruang lingkup dari sistem informasi penjualan yang berorientasi logistik juga wajib mencakup tahapan pengiriman dan *monitoring* barang kepada pihak penerima [15]. Fitur monitoring pergerakan pengiriman di dalam modul antarmuka memberikan kemampuan bagi perusahaan untuk melacak perubahan status logistik armada secara langsung, sehingga konfirmasi pesanan tiba dapat divalidasi oleh sistem tanpa perlu lagi mengandalkan komunikasi pertukaran pesan secara manual kepada pihak pengemudi atau pelanggan [16].

Aspek pamungkas yang menyempurnakan alur kerja sistem informasi ini adalah pengelolaan modul pembayaran pesanan [17]. Sentuhan digitalisasi pada tahapan pendataan pembayaran memfasilitasi pencatatan status pelunasan faktur secara terstruktur di dalam dasbor rekapitulasi, di mana pimpinan usaha dapat seketika mendeteksi pesanan apa saja yang kewajibannya sudah tuntas dan mana yang masih berstatus piutang menunggak, sehingga perputaran kas finansial operasional tetap terukur dan akuntabel [18].

## 2.3 Aplikasi Berbasis *Mobile*

Aplikasi adalah suatu perangkat lunak (*software*) terprogram yang dirancang dengan mematuhi serangkaian instruksi bahasa pemrograman tertentu untuk memproses data komputasi sesuai dengan kehendak pengembangnya[19]. Pada praktiknya, aplikasi berfungsi sebagai jembatan alat komputasi yang siap pakai untuk menjembatani interaksi manusia (*brainware*) guna menyelesaikan berbagai instruksi tugas administratif secara terpusat lewat visual antarmuka layar [20].

Sementara itu, aplikasi *mobile* diartikan sebagai bentuk khusus dari rekayasa pengembangan perangkat lunak yang dirakit dan dioptimalkan secara spesifik agar mampu dioperasikan dengan stabil di atas perangkat bergerak (*mobile devices*), seperti gawai ponsel pintar (*smartphone*) dan komputer tablet [21] Berbeda halnya dengan sistem yang terpaku

pada layar komputer meja (*desktop*), platform berarsitektur *mobile* dibangun agar sepenuhnya kompatibel terhadap layar sentuh dan mampu menyerap sumber daya memori fisik genggaman pengguna [22]. Karakteristik keunggulan paling menonjol yang menjadikan aplikasi *mobile* amat diminati di sektor industri adalah tingkat mobilitasnya yang absolut [21].

## 2.4 Black-box Testing

*Blackbox testing* merupakan salah satu metode pengujian validasi perangkat lunak yang murni difokuskan pada penilaian fungsionalitas antarmuka dan spesifikasi alur program, tanpa mengharuskan penguji untuk membedah atau mengenali struktur kerangka kode sumber (*source code*) penyusunnya [23]. Pengujian ini mengibaratkan sistem sebagai sebuah "kotak hitam" di mana penguji hanya berinteraksi melalui antarmuka pengguna untuk memberikan masukan (*input*) dan mengamati apakah keluaran (*output*) yang dihasilkan sudah sesuai dengan kebutuhan fungsional yang diharapkan [23].

Tujuan utama dari metode pengujian ini adalah untuk menemukan berbagai kategori kesalahan fungsional, seperti fungsi yang hilang atau tidak berjalan, kesalahan pada antarmuka (*interface*), kesalahan pada struktur data atau akses basis data eksternal, hingga kesalahan inisialisasi dan terminasi program [24]. Dalam praktiknya, terdapat beberapa teknik turunan yang umum digunakan dalam *black box testing*, di antaranya:

1. *Equivalence Partitioning*: Teknik pengujian yang membagi domain masukan dari suatu program ke dalam kelas-kelas data, di mana dari setiap kelas tersebut dapat ditarik instrumen pengujian yang merepresentasikan kevalidan data [25].
2. *Boundary Value Analysis*: Teknik pengujian yang berfokus pada pengujian nilai batas (*boundary*) dari suatu rentang data masukan, karena kesalahan program lebih sering terjadi pada nilai batas (minimum atau maksimum) dibandingkan pada nilai tengah [25].

Melalui penerapan *black box testing*, pengembang dapat memastikan bahwa aplikasi yang dibangun, baik dari sisi fungsionalitas pengiriman maupun pembayaran, dapat berjalan dengan stabil dan merespons interaksi pengguna secara presisi sebelum perangkat lunak tersebut dirilis ke tahap produksi.

## 2.5 Metode Penelitian

Metode penelitian secara teoretis didefinisikan sebagai kerangka kerja ilmiah dan sistematis yang digunakan oleh peneliti untuk mengumpulkan, mengolah, serta menganalisis data guna memecahkan suatu permasalahan yang telah dirumuskan[26]. Dalam konteks ilmu komputer dan rekayasa perangkat lunak, metode penelitian tidak hanya berfokus pada pendekatan filosofis keilmuan, tetapi juga mencakup serangkaian prosedur teknis untuk merancang dan membangun sebuah sistem informasi yang mampu menjawab kebutuhan operasional suatu entitas atau organisasi [27]. Penggunaan metode penelitian yang terstruktur memastikan bahwa perangkat lunak yang dikembangkan memiliki landasan empiris yang kuat, tepat sasaran, dan bukan sekadar asumsi pengembang semata.

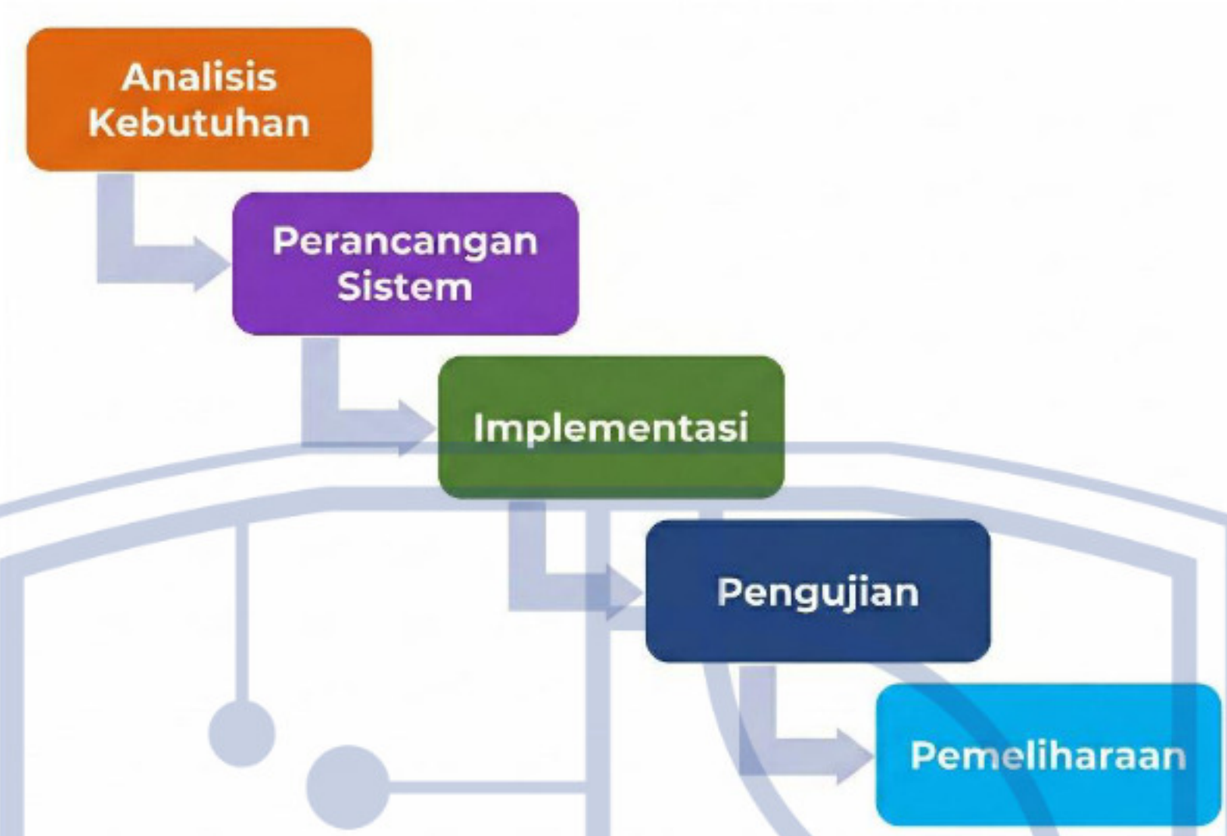
Untuk mendukung tahap analisis dan perancangan sistem dengan baik, diperlukan metode pengumpulan data yang valid dan relevan. Berdasarkan literatur pengembangan sistem informasi, terdapat tiga teknik utama yang lazim digunakan untuk memperoleh spesifikasi kebutuhan perangkat lunak dari lapangan, yaitu observasi, wawancara, dan studi pustaka [28]. Penjabaran teoretis dari masing-masing metode pengumpulan data tersebut adalah sebagai berikut:

1. Observasi (Pengamatan Langsung): Pengamatan langsung terhadap aktivitas operasional di lapangan guna memetakan alur kerja (*workflow*) sistem berjalan dan mengidentifikasi kendala teknis secara empiris [28].
2. Wawancara (*Interview*): Proses interaksi dan tanya jawab dengan pemangku kepentingan untuk menggali spesifikasi kebutuhan sistem, ekspektasi pengguna, dan aturan bisnis secara mendalam [28].
3. Studi Pustaka (*Literature Review*): Pengkajian literatur ilmiah dan referensi teknis sebagai data sekunder untuk membangun landasan teoretis serta menentukan acuan metode pengembangan sistem [28].

## 2.6 Metode Pengembangan Perangkat Lunak

Penelitian ini menggunakan model *Waterfall* sebagai metode pengembangan perangkat lunak (SDLC - *Software Development Life Cycle*). Model *Waterfall* merupakan pendekatan klasik yang bersifat linier dan sekuensial, di mana setiap fase dalam pengembangan perangkat lunak harus diselesaikan secara utuh sebelum beranjak ke fase berikutnya. Pemilihan metode ini didasarkan pada karakteristik alur kerjanya yang

terstruktur, dokumentasi yang rapi, dan kemampuannya untuk meminimalisir perubahan spesifikasi secara mendadak di tengah proses pengerjaan [29].




Gambar 2.1 Metode Waterfall

Tahapan pengembangan aplikasi menggunakan metode *Waterfall* pada penelitian ini dijabarkan sebagai berikut:

1. Analisis Kebutuhan (*Requirements Analysis*): Tahap pengumpulan dan pendokumentasian spesifikasi perangkat lunak untuk mendefinisikan fungsi, batasan, dan kapabilitas sistem yang akan dibangun [30].
2. Perancangan Sistem (*System Design*): Proses menerjemahkan dokumen kebutuhan ke dalam rancangan teknis, meliputi arsitektur sistem, pemodelan logika, struktur basis data, dan antarmuka pengguna [31].
3. Implementasi (*Implementation*): Tahap penerjemahan rancangan desain ke dalam baris kode program (*coding*) untuk membangun perangkat lunak yang fungsional secara utuh [31].
4. Pengujian (*Verification/Testing*): Proses evaluasi menyeluruh pada unit program untuk mendeteksi kecacatan (*bug*) dan memastikan sistem beroperasi sesuai dengan spesifikasi awal [25].
5. Pemeliharaan (*Maintenance*): Tahap pasca-rilis yang mencakup perbaikan kesalahan yang mungkin tersisa, optimalisasi kinerja, serta penyesuaian sistem terhadap perubahan kebutuhan di masa mendatang [30].

## 2.7 Flutter

*Flutter* dikenal sebagai sebuah kerangka kerja antarmuka (*UI framework*) bersifat sumber terbuka (*open-source*) yang dirintis oleh *Google*, difokuskan untuk merancang aplikasi seluler hibrida (*hybrid*) berkinerja unggul. Berdiri di atas pilar bahasa pemrograman *Dart*, kerangka kerja komprehensif ini menawarkan kemampuan bagi pengembang perangkat lunak untuk merakit satu basis kode (*single codebase*) dan menjalankannya secara optimal tanpa kendala di atas ekosistem sistem operasi *Android* maupun *iOS* [32]. Kemudahan integrasi dan adaptasi *cross-platform* yang dibawanya menjadikan teknologi *Flutter* sebagai standar populer terbaru bagi industri pengerjaan aplikasi gawai pintar masa kini.

Keunggulan dominan yang melekat pada arsitektur pemrograman *Flutter* terletak pada kekayaan perbendaharaan elemen visual (*widget*) yang sangat adaptif serta bisa dimodifikasi murni sesuai cetak biru rancangan antarmuka aplikasi. Di samping itu, penyematan fitur pengawasan langsung bernama *Hot Reload* sukses memangkas secara tajam siklus kompilasi waktu pengembangan (*development time*), mengingat perubahan baris kode apa pun akan terpantul secara langsung seketika pada layar perangkat uji coba (*emulator*) [33].

## 2.8 Golang

*Go*, atau yang lebih populer dikenal sebagai *Golang*, adalah bahasa pemrograman bersifat sumber terbuka (*open-source*) yang dikembangkan oleh tim insinyur *Google* untuk menciptakan perangkat lunak yang andal dan efisien. Bahasa ini dirancang sebagai bahasa yang dikompilasi (*compiled language*) dan diketik secara statis (*statically typed*), namun memiliki sintaksis yang ringkas sehingga mudah dibaca layaknya bahasa dinamis. Dalam arsitektur pengembangan sistem modern, *Golang* sering menjadi pilihan utama untuk membangun sisi *back-end* atau *server-side* karena kemampuannya dalam menangani konkurensi (proses yang berjalan bersamaan) dengan konsumsi memori yang jauh lebih rendah dibandingkan bahasa pemrograman lain [34].

Keunggulan utama yang ditawarkan oleh *Golang* terletak pada fitur *Goroutines*, yaitu mekanisme manajemen *thread* yang sangat ringan, memungkinkan sebuah aplikasi untuk melayani ribuan permintaan (*request*) secara simultan tanpa membebani kinerja *server*. Efisiensi ini menjadikan Golang sangat relevan untuk diterapkan dalam pengembangan *Application Programming Interface* (*API*) yang membutuhkan respon cepat

dalam pertukaran data antara aplikasi *mobile* dan basis data pusat [35]. Selain itu, ekosistem *Golang* yang kuat mendukung proses pengembangan yang lebih terstruktur, meminimalisir *bug* pada saat fase produksi (*runtime*), serta mempercepat proses kompilasi kode program.

## 2.9 Postgre SQL

*PostgreSQL* adalah sistem manajemen basis data relasional objek (*Object-Relational Database Management System* atau ORDBMS) yang bersifat sumber terbuka dan dikenal luas karena stabilitas serta kepatuhannya terhadap standar teknis *SQL*. Sebagai basis data tingkat lanjut, *PostgreSQL* dirancang untuk menangani beban kerja mulai dari aplikasi tunggal hingga layanan *web* berskala besar dengan banyak pengguna yang melakukan akses data secara bersamaan. Kemampuannya dalam menjaga integritas data (*ACID-compliant*) menjadikannya solusi penyimpanan yang sangat andal untuk sistem yang membutuhkan akurasi tinggi, seperti pencatatan transaksi keuangan dan riwayat logistik barang [36].

Selain keandalannya dalam menjaga konsistensi data, *PostgreSQL* juga memiliki fleksibilitas tinggi dalam menangani berbagai tipe data, termasuk dukungan terhadap format JSON (*JavaScript Object Notation*) yang lazim digunakan dalam komunikasi aplikasi *mobile*. Fitur ini memungkinkan pengembang untuk menyimpan data semi-terstruktur tanpa harus mengubah skema basis data secara drastis, sehingga mempercepat proses adaptasi sistem terhadap perubahan kebutuhan bisnis [37]. Dukungan komunitas yang besar dan performa *query* yang optimal menjadikan *PostgreSQL* pilihan strategis untuk memastikan data pengiriman dan pembayaran dapat diakses dengan cepat dan aman.