BAB II
DASAR TEORI

2.1 Sistem Informasi

Sistem informasi dapat didefinisikan sistem yang terstruktur dan dirancang untuk mengolah informasi secara efektif dengan tujuan tertentu. Informasi yang dihasilkan oleh sistem haruslah berguna dan relevan dengan kebutuhan, sehingga dapat disampaikan dengan jelas kepada penerima dan memanfaatkannya untuk mencapai tujuan organisasi[3]. Selain membantu dalam menjalankan tugas-tugas rutin sehari-hari, sistem informasi juga berperan penting dalam meningkatkan efisiensi operasional organisasi. Dengan pengolahan data yang akurat dan tepat waktu, sistem ini mampu menyediakan laporan yang dibutuhkan oleh berbagai tingkatan manajemen, yang kemudian menjadi dasar bagi pengambilan keputusan guna mendukung pencapaian tujuan organisasi secara keseluruhan[4].

Dalam penelitian ini, dikembangkan sistem informasi persediaan yang relevan dengan kebutuhan utama, seperti memantau stok barang, memfasilitasi penjualan, dan mengoptimalkan persediaan. Sistem ini dirancang untuk memenuhi kebutuhan spesifik pengguna dengan mencakup fitur-fitur seperti perhitungan, pemantauan, dan pelaporan persediaan, serta pengelolaan informasi terkait persediaan barang.[5].

2.2 Rekayasa Perangkat Lunak

Perangkat lunak adalah hasil dari pendekatan sistematis dalam pengembangan, operasi, dan pemeliharaan, yang dikenal sebagai rekayasa perangkat lunak. Pemilihan model proses dalam rekayasa perangkat lunak disesuaikan dengan jenis aplikasi, proyek, metode, alat, pengendalian, dan penyampaian yang dibutuhkan. Beberapa model proses yang umum digunakan meliputi model sekuensial linier (*waterfall*), prototipe, RAD, inkremental, spiral, pengembangan terpadu, metode formal, dan teknik generasi keempat, masing-masing dengan pendekatan dan keunggulan khusus[6]. Berikutnya, metode pengembangan sistem yang akan diterapkan.

2.2.1 Metode *Prototype*

Model *prototyping* adalah cara untuk mengumpulkan informasi tentang apa yang diinginkan pengguna dari suatu perangkat lunak. Berfokus pada bagaimana perangkat lunak akan terihat dan berfungsi bagi pengguna. *Prototype* dibuat, kemudian dievaluasi oleh pengguna, dan digunakan untuk mengetahui kebutuhan pengembangan lebih lanjut. *Prototype* memberikan gambaran kepada pembuat dan pengguna tentang bagaimana sistem akan berjalan, dan proses pembuatannya disebut *prototyping*. *Prototype* awal akan digunakan untuk mengindentifikasi

masalah dan mencari solusi sebelum pengembangan lebih lanjut. Metode *prototyping* digunakan untuk membuat gambaran awal aplikasi, dimana versi awal dievaluasi oleh pengguna dan menjadi panduan bagi pengembang dalam merancang aplikasi[7].

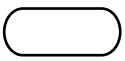
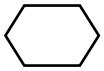
Setelah metode *prototyping* digunakan untuk mengumpulkan informasi dan membentuk gambaran awal aplikasi, langkah selanjutnya adalah menerjemahkan kebutuhan dan fungsi ke dalam bentuk yang lebih teknis dan terstruktur.

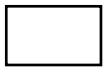
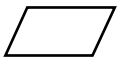
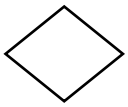
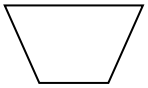
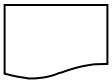
2.2.1.1 Flowchart

Flowchart adalah cara untuk menggambarkan langkah-langkah atau urutan tindakan dari awal hingga akhir dalam suatu program atau sistem. *Flowchart* sistem menjelaskan cara kerja sistem dengan menunjukkan *input*, *output* dan media yang digunakan untuk penyimpanan data dalam prosesnya. Sementara *flowchart* program adalah diagram yang menggunakan simbol-simbol khusus untuk menunjukkan langkah-langkah secara detail dan bagaimana setiap langkah terhubung satu sama lain dalam suatu program. Dalam pembuatan *flowchart* dapat beberapa petunjuk yang harus diperhatikan:

1. *Flowchart* dibuat dari atas ke bawah dan dari kiri ke kanan mengikuti prosesnya.
2. Aktivitas yang digambarkan sebaiknya dapat didefinisikan dan dapat dipahami oleh pembaca.
3. Setiap aktivitas dimulai dan diakhiri harus ditentukan secara jelas.
4. Langkah-langkah setiap aktivitas yang digambarkan harusnya dapat dijelaskan dengan kata kerja.
5. Setiap langkah-langkah aktivitas harus sesuai urutannya.
6. Pastikan batasan aktivitas terdefinisi dengan jelas. Jika ada percabangan yang tidak terkait dengan sistem, bisa diletakkan pada halaman yang terpisah dengan simbol konektor.
7. Menggunakan symbol-simbol *flowchart* yang standar. Simbol *flowchart* ditunjukkan pada tabel 2.1[8].

Tabel 2. 1 Simbol-simbol *Flowchart*

NO	SIMBOL	NAMA	FUNGSI
1.		*TERMINATOR*	Permulaan atau akhir program
2.		GARIS ALIR (*FLOW LINE*)	Arah aliran program
3.		*PREPARATION*	Proses inisialisasi/pemberian harga awal

4.		PROSES	Proses perhitungan/proses pengolahan data
5.		*INPUT/OUTPUT DATA*	Proses *input/output* data, parameter, dan informasi
6.		*PREDEFINED PROCESS* (SUB PROGRAM)	Permulaan sub program/proses menjalankan sub program
7.		*DECISION*	Perbandingan pernyataan, penyeleksian data yang memberikan pilihan untuk langkah selanjutnya
8.		*ON PAGE CONNECTOR*	Penghubung bagian-bagian *flowchart* yang berada pada suatu halaman
9.		*OFF PAGE CONNECTOR*	Penghubung bagian-bagian *flowchart* yang berada pada halaman berbeda
10.		*MANUAL OPERATION*	Langkah manual seperti verifikasi dokumen dan pengecekan fisik barang yang belum terotomatisasi oleh sistem.
11.		*DOCUMENT*	Menunjukkan langkah yang menghasilkan dokumen atau laporan, baik fisik maupun digital.

2.2.1.2 Unifed Modelling Language (UML)

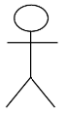
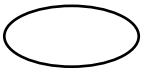
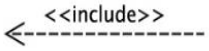
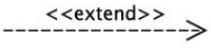
UML adalah bahasa pemodelan yang populer dengan visualisasi sistem dan dokumentasi yang baik. UML dapat membantu mendeskripsikan dan merancang sistem perangkat lunak, terutama yang dibangun dengan pemrograman berorientasi objek. UML berkembang dari penggabungan berbagai bahasa pemodelan grafis berorientasi objek pada akhir 1980-an dan awal 1990-an. Meskipun UML paling sering digunakan dalam metodologi berorientasi objek,

penggunaannya tidak terbatas pada metodologi tertentu[9]. Jenis *Diagram Unified Model Language* (UML) yang digunakan dalam penelitian ini yaitu:

A. *Use Case Diagram*

Use case adalah cara untuk menggambarkan fungsi yang diberikan oleh sistem melalui dialog antara aktor dan sistem. *Use case* merupakan pola atau bentuk perilaku yang menunjukkan bagaimana sistem berinteraksi dengan penggunanya. Setiap *use case* menggambarkan serangkaian transaksi atau proses yang terkait antara aktor (pengguna) dan sistem dalam sebuah dialog[10]. Dua elemen penting dalam diagram *use case* ialah aktor dan *use case*. Aktor ialah yang merupakan orang yang secara langsung berinteraksi dengan sistem, sedangkan *use case* yang menggambarkan bagaimana aktor tersebut berinteraksi dengan sistem[11]. Diagram *use case* memiliki simbol-simbol yang dapat dilihat pada tabel 2.2.

Tabel 2. 2 Simbol-simbol *Use Case*

No	Simbol	Keterangan
1.		Pengguna yang berinteraksi dengan sistem. Aktor yang ditunjukkan dengan nama dan perannya dalam sistem
2.		Gambaran dari interaksi antara aktor dan sistem
3.		Menggambarkan interaksi aktor dan *use case* secara langsung
4.		Menggambarkan fungsi sebuah *use case* yang hanya dapat dipenuhi dengan bantuan dari *use case* yang lainnya
5.		Menggambarkan fungsi sebuah *use case* yang dapat diperluas oleh *use case* lain, jika dibutuhkan
6.		Relasi antara suatu aktor dengan aktor turunannya

2.3 PBO (Pemrograman Berorientasi Obyek)

Pemrograman berorientasi objek atau *Object-Oriented Programming* (OOP) mempermudah pendekatan pemrograman yang menggunakan objek dan kelas[12]. OOP mempermudah pembuatan program dengan berbagai keuntungan, antara lain:

1. *Reusability*: Kode yang dibuat dapat digunakan kembali, sehingga mengurangi duplikasi dan meningkatkan efisiensi pengembangan.
2. *Extensibility*: Pemrograman dapat membuat metode baru atau mengubah metode yang sudah ada tanpa harus membuat kode dari awal, memungkinkan fleksibilitas dalam pengembangan.
3. *Maintainability*: Kode yang dibuat lebih mudah dikelola, terutama dalam aplikasi berskala besar. OOP menggunakan konsep modularitas, yang membantu dalam menangani potensi kesalahan selama pengembangan.

Dalam OOP, objek adalah unit terkecil yang memiliki data (karakteristik) dan fungsi. Objek mewakili entitas yang memiliki keadaan, perilaku, dan identitas, serta dideklarasikan sebagai *instance* dari sebuah kelas. Kelas adalah wadah yang mendeskripsikan karakteristik dan fungsi objek, dan harus diciptakan terlebih dahulu sebelum objeknya. Sebagai salah satu bahasa pemrograman berorientasi objek, Jawa memiliki beberapa konsep penting dalam OOP:

1. *Encapsulation*: Proses membungkus data dan metode yang menyusun kelas sehingga kelas dipandang sebagai suatu modul.
2. *Inheritance* (Pewarisan): Proses pewarisan data dan metode dari satu kelas ke kelas lain. Semua data dan metode dari kelas asal diwariskan ke kelas baru.
3. *Polymorphism*: Konsep di mana suatu modul dapat memiliki banyak bentuk. Dalam OOP, ini berarti memiliki kesamaan nama namun dengan perilaku yang berbeda, sehingga implemtasi kode juga berbeda.

2.4 *Database*

Database adalah sistem yang dirancang untuk menyimpan data secara teratur dan mudah diakses. *Database* terdiri dari kumpulan data yang terstruktur untuk satu atau bahkan lebih penggunaan, biasanya dalam format digital. Manajemen database dilakukan melalui *software* yang disebut *Database Management System* (DBMS). DBMS bertanggung jawab atas penyimpanan dan pengelolaan data dalam *database*, serta membantu dalam membuat, memperbarui, mencari, dan mengakses data. Beberapa sistem manajemen *database* yang umum digunakan saat ini antara lain *MySQL*, *Oracle*, dan *PostgreSQL*[13]. Beberapa fungsi *database* meliputi:

1. Menyusun data agar mudah dikenali, seperti dengan menggunakan tabel atau kolom yang berbeda.
2. Mengurangi duplikasi data.
3. Mempermudah pengguna dalam melakukan input, ubah, dan hapus data.
4. Penggunaan penyimpanan digital.

   Beberapa jenis *database* meliputi:

   1. *Operational Database*, yang digunakan untuk menyimpan detail data yang sangat rinci dan mudah diakses, seringkali digunakan untuk data pelanggan.
   2. *Relational Database*, memungkinkan pengguna untuk mengakses atau mencari informasi dari tabel yang berbeda.
   3. *Distributed Database*, digunakan untuk menyebarluaskan data secara luas tetapi tetap terhubung dan dapat diakses secara bersamaan.
   4. *External Database*, digunakan untuk tujuan bisnis dan tidak memberikan akses mudah kepada publik.

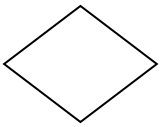
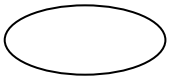
Terdapat pula diagram ERD yang dapat digunakan sebagai alat bantu dalam proses pembuatan *database*.

2.4.1 ERD

Entity Relationship Diagram (ERD) adalah sebuah penggambaran model data dengan menggunakan gambaran grafis. Tujuannya adalah untuk membantu dalam pengembangan sistem dengan menyediakan detail informasi yang diperlukan. ERD mempermudah analisis sistem dan memenuhi kebutuhan pengembangan sistem[14]. ERD merupakan diagram yang merancang struktur basis data dengan menunjukkan hubungan antara entitas atau objek, beserta atribut-atributnya. Diagram ini membantu merancang basis data relasional dengan menggambarkan tabel, atribut, hingga derajat relasi. Jika ERD dirancang dengan benar, maka basis data yang dibuat berdasarkan ERD tersebut akan memiliki struktur yang tepat dan sesuai dengan kebutuhan sistem[15]. Simbol-simbol yang digunakan pada ERD dapat dilihat pada tabel 2.3 [16]:

Tabel 2. 3 Simbol-simbol ERD

No	Simbol	Keterangan
1.		Entitas, adalah suatu objek yang dapat diindentifikasi dalam lingkungan pemakai.

2.		Relasi, menunjukkan adanya hubungan diantara sejumlah entitas yang berbeda.
3.		Atribut, berfungsi mendeskripsikan karakter entitas (atribut yang berfungsi sebagai key diberi garis bawah).
4.		Garis, sebagai penghubung antara relasi dengan entitas, relasi dan entitas dengan atribut.

2.5 Persediaan

Persediaan merupakan kumpulan barang yang disimpan oleh suatu perusahaan dengan tujuan untuk mendukung operasional bisnis. Barang-barang disiapkan baik untuk dijual kepada pelanggan maupun digunakan dalam proses produksi untuk menjaga kelancaran aktivitas bisnis perusahaan. Persediaan memainkan peran penting sebagai salah satu sumber utama pendapatan perusahaan, karena ketersediaan barang yang memadai dan tepat waktu dapat memenuhi permintaan pelanggan, serta memastikan proses produksi berjalan tanpa hambatan. Dengan demikian, pengelolaan persediaan yang efektif menjadi kunci dalam menjaga keberlanjutan operasional dan pendapatan perusahaan[17].

2.5.1 Metode FIFO (*First In First Out*)

First-in First-Out (FIFO) adalah metode yang digunakan untuk menyelesaikan masalah dengan cara menjual atau menggunakan barang yang pertama kali masuk terlebih dahulu. Ini berarti barang yang pertama kali masuk ke dalam gudang akan dianggap sebagai barang yang pertama kali terjual atau digunakan, sehingga memastikan bahwa barang tersebut terjual atau digunakan dalam keadaan yang baik. FIFO sering digunakan dalam berbagai situasi dalam kehidupan sehari-hari, aplikasi, dan teknologi. FIFO adalah algoritma yang sederhana dan mengikuti urutan masuknya barang, sehingga barang diproses sesuai dengan urutan tersebut[18].

Metode FIFO bisa digunakan dalam berbagai kegiatan seperti manajemen persediaan di gudang, pengelolaan stok makanan di restoran, dan distribusi barang di retail. Selain itu, FIFO juga diterapkan dalam manajemen produk di industri farmasi untuk memastikan obat-obatan yang pertama kali diproduksi digunakan terlebih dahulu sebelum kadaluwarsa. Barang-barang

seperti bahan makanan, produk farmasi, suku cadang elektronik, dan berbagai jenis bahan mentah sering diatur menggunakan metode ini untuk menjaga kualitas dan menghindari pemborosan[19].