BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

BAB II
DASAR TEORI

2.1. TinjauanPustaka

2.2. Landasan Teori

Dalam penelitian ini diperlukan adanya teori yang mendasar dalam menunjang proses penelitian, berikut antara lain :

2.2.1. Sistem Informasi

Sistem merupakan suatu kesatuan yang bersifat utuh yang didalamnya terdapat sekumpulan bagian sistem baik dalam bentuk abstrak maupun terintegrasi yang digandakan secara fisik untuk bekerja sama agar mencapai suatu tujuan[2]. Sistem memiliki beberapa karakteristik atau sifat tertentu, yaitu yang terdiri dari :

a) Memiliki komponen sistem (*Component*)
   Suatu sistem terdiri dari beberapa komponen dimana komponen tersebut saling terhubung dan berinteraksi dan juga bekerja sama menghasilkan suatu komponen sistem.
b) Memiliki batasan sistem (*Bondry*)
   Suatu daerah yang membatasi sesuatu dalam sebuah sistem dengan sistem yang lain agar tidak terlalu luas cakupan suatu sistem.
c) Memiliki penghubung sistem (*Interface*)
   Suatu media penghubung antara suatu bagian sistem dengan bagian sistem yang lain karena bertujuan untuk memungkinkan berbagai aliran suatu sumber daya mengalir dari suatu bagian sistem ke bagian sistem lainnya.
d) Memiliki sasaran sistem (Objek)
   Sebuah objek yang bertujuan untuk mencapai sebuah tujuan yang diinginkan oleh sistem dan dikatakan berhasil ketika sistem tersebut mengenai sasaran atau tujuan.

2.2.2. *Monitoring*

Monitoring atau sebuah penelitian yang diartikan sebagai penelitian yang memperlajari kegiatan yang dilakukan dengan mengedepankan kesesuaian sebuah rencana, suatu permasalahan yang dihadapi dan menilai ketepatan pola kerja dan manajemen dalam mencapai sebuah tujuan dan pengetahuan tentang indikator kemajuan antar suatu kegiatan karena pemantauan menjadi sebuah proses yang penting untuk memastikan suatu kegiatan berjalan sesuai rencana dan dapat mengetahui tercapai atau tidaknya suatu tujuan[3]. Tujuan *Monitoring*[4] :

1. Menilai apakah suatu kegiatan pemantauan yang direcanakan dilakukan secara konsisten sesuai dengan rencana awal.
2. Mengamati dan memantau semua aktivitas suatu proses pemantauan yang terkait dengan objek program.
3. Mengidentifikasi potensi masalah sehingga dapat diselesaikan dengan cepat.

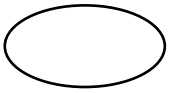
2.2.3. Absensi

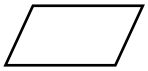
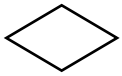
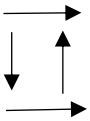
Absensi adalah suatu aktivitas penting yang dilakukan oleh individu di dalam sebuah instansi atau organisasi dengan tujuan utama untuk membuktikan kehadiran mereka dalam suatu pertemuan atau kegiatan resmi. Proses ini berkaitan erat dengan disiplin dan tanggung jawab yang ditetapkan oleh masing-masing perusahaan atau institusi. Kehadiran yang teratur dan tepat waktu sering kali menjadi indikator penting dalam mengevaluasi kinerja, komitmen, dan kedisiplinan anggota organisasi. Dengan demikian, absensi tidak hanya berfungsi sebagai catatan kehadiran, tetapi juga sebagai alat untuk mengukur tingkat kedisiplinan dan tanggung jawab individu terhadap kewajiban mereka.[5].

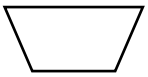
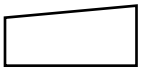
2.2.4. *Flowchart*

Bagan alir (*Flowchart*) adalah teknik analisis bergambar yang digunakan untuk menggambarkan berbagai aspek dari sistem informasi dengan cara yang jelas, ringkas, dan logis. Bagan alir mencatat bagaimana proses bisnis dijalankan dan bagaimana dokumen mengalir melalui organisasi. *Flowchart* adalah representasi visual dari sistem, prosedur, dan pengendalian internal yang telah diterapkan oleh perusahaan. *Flowchart* menggambarkan langkah-langkah dan urutan prosedur suatu program secara grafis. Biasanya, Flowchart mempermudah penyelesaian masalah yang memerlukan studi dan evaluasi lebih lanjut[6]. Berikut ini symbol-simbol yang digunakan dalam *Flowchart* dapat dilihat pada Tabel 2.1.

Tabel 2.1 Simbol dan Fungsi *Flowchart*

No.	Simbol	Nama	Keterangan
1.		Terminal	Simbol oval atau *elips* digunakan untuk menunjukkan titik awal atau akhir dari suatu proses atau *algoritma*, menandai di mana

			Flowchart dimulai dan berakhir.
2.		Process	Simbol yang menunjukan pengolahan data yang dilakukan oleh komputer.
3.		Output Input	Simbol jajaran genjang digunakan untuk menunjukkan langkah di mana data dimasukkan ke dalam sistem (*input*) atau dikeluarkan dari sistem (*output*), seperti menerima input pengguna atau menampilkan hasil.
4.		Decision	Simbol belah ketupat (*diamond*) menunjukkan titik di mana keputusan harus dibuat, biasanya menghasilkan dua atau lebih jalur aliran proses, seperti "ya" atau "tidak".
5.		Flow	Simbol panah digunakan untuk menunjukkan arah aliran proses atau urutan langkah-langkah dalam *Flowchart*, menghubungkan berbagai simbol lainnya.
6.		Document	Simbol persegi panjang dengan bagian bawah bergelombang menunjukkan langkah yang menghasilkan dokumen atau laporan, baik fisik maupun digital.

7.		Manual Operation	Simbol trapesium menunjukkan langkah yang dilakukan secara manual oleh manusia, seperti verifikasi dokumen atau pengecekan fisik barang yang tidak diotomatisasi oleh sistem.
8.		Manual Input	Simbol trapesium miring menunjukkan titik di mana pengguna memasukkan data secara manual, seperti mengisi formulir kertas atau entri data ke dalam sistem.

2.2.5. *Laravel*

Laravel adalah *framework* PHP yang populer untuk pengembangan aplikasi web, dikenal karena sintaksnya yang bersih dan elegan, sehingga mudah dipahami dan digunakan. *Framework* ini memudahkan pengembang untuk menulis kode yang lebih ekspresif dan gampang dipelajari. Dengan menggunakan arsitektur *Model-View-Controller* (MVC), *Laravel* memisahkan logika bisnis dari tampilan, yang mempermudah pengembangan dan pemeliharaan kode. *Laravel* juga dilengkapi dengan *Blade Templating Engine*, yaitu mesin template yang memungkinkan pengembang membuat tampilan dengan sintaks yang rapi dan ekspresif, serta menyediakan fitur seperti pewarisan (*inheritance*) dan kontrol struktural yang kuat. Selain itu, *Laravel* memiliki dukungan komunitas yang besar dan aktif, dengan banyak sumber daya *online*, forum, dan paket tambahan yang membantu pengembang mengatasi masalah atau mendapatkan bantuan saat diperlukan[7].

2.2.6. OOP (*Object Orieted Programing*)

Pemrograman berorientasi objek, atau *object-oriented programming* (OOP), adalah pendekatan pemrograman yang menggunakan objek dan kelas. OOP memudahkan dalam pembuatan program dengan memberikan beberapa keuntungan, antara lain: 1) *Reusability*, yaitu kemampuan untuk menggunakan kembali kode yang sudah dibuat, 2) *Extensibility*, di mana pengembang dapat menambahkan metode baru atau mengubah yang sudah ada sesuai kebutuhan

tanpa harus menulis ulang kode dari awal, dan 3) *Maintainability*, yaitu kemudahan dalam mengelola kode, terutama dalam aplikasi berskala besar, karena OOP sudah menerapkan konsep modularitas yang membantu mengatasi potensi kesalahan selama pengembangan[8].

2.2.7. Basis Data

Basis data adalah kumpulan elemen data logis yang terintegrasi dan saling berhubungan, yang mengonsolidasi catatan yang sebelumnya disimpan dalam file terpisah. Basis data dirancang untuk memenuhi kebutuhan informasi organisasi dan dapat diakses oleh banyak pengguna. Data disimpan di media penyimpanan eksternal dan dimanipulasi oleh perangkat lunak khusus. Sebagai komponen penting dalam sistem informasi, basis data menyediakan informasi esensial bagi penggunanya dan mempermudah serta mempercepat pemanggilan serta pemanfaatan data. Sistem basis data mengintegrasikan data yang saling berhubungan dan membuatnya tersedia untuk berbagai aplikasi dalam organisasi, mendukung proses pengambilan keputusan dengan menyediakan informasi yang optimal[9].

2.2.8. DBMS (*Database Management System*)

DBMS (*Data Base Management System*) adalah perangkat lunak yang digunakan untuk mengelola basis data. DBMS bertugas untuk menyimpan, mengambil, memperbarui, dan menghapus data dalam basis data. DBMS menyediakan antarmuka dan alat untuk mengatur struktur basis data, memanipulasi data, serta menjaga keamanan dan *integritas* data. Sebagai alternatif, DBMS dapat didefinisikan sebagai sistem pencatatan terkomputerisasi yang menyimpan informasi dan memungkinkan pengguna untuk menambah, menghapus, memodifikasi, mengambil, dan memperbarui informasi tersebut[10]. DBMS menyediakan berbagai fasilitas yaitu :

1. DDL (*Data Definition Language*) merupakan bagian dari bahasa query yang digunakan dalam *database* untuk mendefinisikan struktur dan skema *database*. DDL digunakan untuk membuat, mengubah, dan menghapus objek-objek *database* seperti tabel, indeks, tampilan, dan konstrain. Dengan perintah-perintah DDL, pengguna dapat mendefinisikan entitas dan hubungan antar entitas dalam database. Beberapa perintah DDL yang umum termasuk :
   a. *CREATE*: Untuk membuat objek baru dalam database, seperti tabel, indeks, atau tampilan.
   b. *ALTER*: Untuk mengubah struktur objek yang sudah ada dalam *database*, misalnya menambahkan kolom baru ke tabel.
   c. *DROP*: Untuk menghapus objek dari *database*, seperti tabel atau indeks.

2. DML (*Data Manipulation Language*) merupakan bagian dari bahasa query database yang berfungsi untuk memanipulasi data dalam tabel. DML memungkinkan pengguna untuk menyisipkan (*insert*), memperbarui (*update*), menghapus (*delete*), dan mengambil (*select*) data dari tabel. Dengan menggunakan DML, pengguna dapat mengubah nilai-nilai dalam tabel, mengambil subset data tertentu, dan melakukan berbagai operasi lainnya terkait data. Beberapa perintah DML yang sering digunakan meliputi :
   a. *INSERT*: Untuk menambahkan data baru ke dalam tabel.
   b. *UPDATE*: Untuk mengubah data yang sudah ada dalam tabel.
   c. *DELETE*: Untuk menghapus data dari tabel.
   d. *SELECT*: Untuk mengambil data spesifik dari tabel.

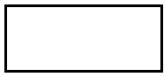
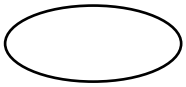
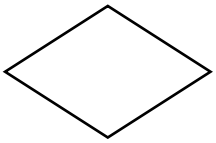
2.2.9. SQL (*Structure Query Language*)

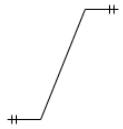
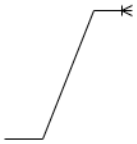
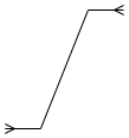
Structured Query Language (SQL) adalah bahasa yang digunakan untuk mengakses data dalam basis data relasional. Bahasa ini telah menjadi standar umum dalam manajemen basis data relasional, dan hampir semua server basis data mendukung SQL untuk keperluan manajemen data. SQL adalah bahasa pemrograman khusus yang digunakan untuk mengelola data dalam RDBMS. SQL terdiri dari perintah-perintah sederhana yang memberikan instruksi untuk manipulasi dan pengambilan data pada *relational database* atau *database* yang terstruktur. Perintah-perintah SQL ini sering disebut sebagai *'query'*[11].

2.2.10. *Entity Relationship Diagram* (ERD)

Entity-Relationship Diagram (ERD) adalah metode pemodelan basis data yang digunakan untuk menggambarkan skema konseptual sistem. Biasanya digunakan dalam sistem basis data relasional dengan pendekatan *top-down*. Diagram yang digunakan untuk representasi model *Entity-Relationship* disebut *Entity-Relationship Diagram* (ERD). *Entity* adalah objek yang dibedakan secara unik dan diidentifikasi, dan hubungannya menghubungkan entitas satu dengan yang lainnya. Atribut menggambarkan karakteristik setiap entitas sesuai dengan konvensi tertentu. Entitas mewakili unit data yang dapat mewakili sesuatu dalam dunia nyata atau abstrak. ERD adalah diagram grafis yang menggambarkan hubungan antara data satu dengan yang lain dalam pembuatan basis data[12]. Berikut ini sImbol-simbol yang digunakan dalam ERD dapat dilihat pada Tabel 2.2.

Tabel 2.2 Simbol dan Fungsi ERD

No.	Simbol	Nama	Keterangan
1.		Entity / Entitas	Entitas adalah objek atau "*thing*" dalam dunia nyata yang dapat diidentifikasi secara unik dan memiliki atribut.
2.		Atribut	Atribut adalah karakteristik atau properti dari suatu entitas yang menjelaskan entitas tersebut.
3.		Multivalue / Atribut multinilai	Atribut multinilai adalah atribut yang dapat memiliki lebih dari satu nilai untuk satu entitas.
4.		Relasi	Relasi adalah hubungan antara dua atau lebih entitas. Relasi menggambarkan bagaimana entitas-entitas tersebut berinteraksi satu sama lain.
5.		Association / Asosiasi	Asosiasi adalah istilah umum untuk hubungan antara entitas dalam diagram ERD, menggambarkan hubungan semantik antara entitas-entitas tersebut.

6.		Relasi Satu ke Satu (One to One)	Relasi satu ke satu adalah relasi di mana satu entitas berhubungan dengan tepat satu entitas lainnya dan sebaliknya.
7.		Relasi Satu ke Banyak (One to Many)	Relasi satu ke banyak adalah relasi di mana satu entitas berhubungan dengan banyak entitas lainnya.
8.		Relasi Banyak ke Banyak (Many to Many)	Relasi banyak ke banyak adalah relasi di mana banyak entitas berhubungan dengan banyak entitas lainnya.

2.2.11. Rekayasa Perangkat Lunak

Perangkat lunak adalah sebuah perintah program dalam sebuah komputer, yang apabila di ekseusi oleh user akan memberikan fungsi dan untuk bekerja seperti yang diharapkan oleh user. Rekayasa Perangkat Lunak (RPL) merupakan proses kegiatan perangkat lunak guna mengembangkan, memelihara, dan membangun kembali dengan menggunakan prinsip rekayasa untuk menghasilkan perangkat lunak yang dapat bekerja lebih efektif dan efisien untuk user [13].

2.2.12. *Waterfall*

Perancangan sistem informasi *Monitoring* kehadiran, peneliti akan menggunakan metode perancangan *waterfall* atau sering dinamakan dengan siklus hidup klasik (*classic life cycle*), model ini mengambarkan sebuah pendekatan yang sistematis serta terstruktur pada pola perancangan perangkat lunak, yang pertama dengan menentukan spesifikasi kebutuhan penggunya lalu melalui tahapan analisis kebutuhan (*requirement*), pemodelan (*design*), penulisan kode (*implementation*), verifikasi dan pengujian (*verification*) dan juga pengembangan sistem (*maintenance*), yang diakhiri dengan dukungan pada perangkat lunak yang dihasilkan[14]. Metode waterfall yang digunakan merupakan metode yang dikembangkan oleh Pressman. Adapun tahapan tahapan metode waterfall antara lain :

a. *Requirement*

Komunikasi antara pengembang sistem sangat penting guna memahami perangkat lunak yang diinginkan oleh pengguna dan batasan yang melekat pada perangkat lunak tersebut. Data dapat diperoleh melalui wawancara, diskusi, atau survei secara langsung. Seluruh informasi yang terkumpul dianalisis dengan teliti untuk memperoleh data yang dibutuhkan oleh pengguna.

b. *Design*

Penulis menciptakan rancangan sistem yang dapat membantu menetapkan komponen fisik (hardware) dan persyaratan sistem serta membantu dalam mengartikulasikan struktur keseluruhan sistem.

c. *Implementation*

Sistem awalnya hanya dibangun di dalam program kecil yang disebut modul, yang kemudian terintegrasi pada tahap berikutnya. Setiap modul dikembangkan dan diuji untuk memastikan fungsionalitasnya dalam apa yang disebut pengujian modul.

d. *Verification*

Proses verifikasi dan pengujian dilakukan untuk menentukan apakah sistem memenuhi persyaratan secara keseluruhan atau sebagian. Pengujian ini dapat dibagi menjadi tiga kategori, yaitu pengujian unit (dilakukan pada modul kode tertentu), pengujian sistem (untuk mengamati respons sistem ketika semua modul terintegrasi), dan pengujian penerimaan (dilakukan dengan atau atas nama pelanggan untuk mengevaluasi kepuasan pelanggan terhadap seluruh kebutuhan).

e. *Maintenance*

Tahap akhir dari model waterfall adalah di mana perangkat lunak yang telah selesai dibuat diimplementasikan dan dipelihara. Pemeliharaan meliputi memperbaiki kesalahan yang tidak terdeteksi pada tahap sebelumnya.

2.2.13. UML (*Unified Modeling Language*)

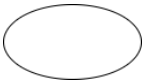
UML (*Unified Modeling Language*) adalah serangkaian diagram yang digunakan untuk melakukan abstraksi terhadap sebuah sistem atau perangkat lunak berbasis objek. UML berfungsi sebagai bahasa visual yang memfasilitasi pemodelan dan komunikasi sistem melalui penggunaan diagram dan teks pendukung. Meskipun UML dirancang khusus untuk pemodelan, penggunaannya tidak terbatas pada satu metodologi tertentu. Namun, dalam praktiknya, UML paling sering diterapkan dalam metodologi yang berorientasi objek. Hal ini karena UML

menawarkan cara yang efektif untuk menggambarkan berbagai aspek dari sistem berbasis objek, mulai dari struktur hingga perilaku, sehingga memudahkan dalam merancang, mendokumentasikan, dan memahami sistem tersebut secara keseluruhan[15].

a) Use Case Diagram

Diagram *use case* adalah alat pemodelan untuk menggambarkan interaksi antara sistem informasi dengan pengguna atau aktor yang terlibat. Ini tidak hanya menunjukkan hubungan antara aktor dan sistem, tetapi juga mengidentifikasi fungsi-fungsi atau aktivitas-aktivitas utama dalam sistem. Setiap *use case* mewakili suatu fungsionalitas atau layanan yang ditawarkan sistem kepada pengguna atau aktor yang berinteraksi dengannya. *Diagram* ini membantu dalam memahami bagaimana aktor berinteraksi dengan sistem dan menentukan hak akses mereka terhadap fungsi-fungsi sistem[16]. Berikut ini simbol-simbol yang digunakan dalam *Use Case Diagram* dapat dilihat pada Tabel 2.3.

Tabel 2.3 Simbol dan Fungsi *Use Case Diagram*

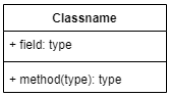
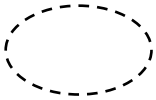
No	Simbol	Nama	Keterangan
1.		Use Case	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor. Biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *use case.*
2.		Aktor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri. Jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang, biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.

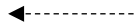
4.		Association	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
5.		Include	Relasi *use case* tambahan ke sebuah use case dimana *use case* yang ditambahkan memerlukan ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case.*

b) Class Diagram

Diagram kelas (*Class Diagram*) adalah salah satu deskripsi yang paling penting dan paling sering digunakan dalam sistem berbasis objek. Diagram kelas memperlihatkan struktur statis dari kelas-kelas inti yang membentuk sistem. Dalam diagram ini, atribut dan metode dari setiap kelas ditampilkan, serta relasi yang ada di antara kelas-kelas tersebut[17]. Berikut ini merupakan simbol-simbol yang digunakan dalam class diagram yang dapat dilihat pada tabel 2.4.

Tabel 2. 4 Simbol dan Fungsi Class Diagram

No	Simbol	Nama	Keterangan
1.		Generalization	Hubugan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
2.		Nary Association	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3.		Class	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
4.		Collaboration	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan

			suatu hasil yang terstruktur bagi suatu aktor.
5.		Realization	Operasi yang benar-benar dilakukan oleh suatu objek.
6.		Dependency	Hubungan dimanan perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen yang bergantung padanya elemen yang beregantung padanya merupakan elemen yang tidak mendiri.
7.		Association	Apa yang menghubungkan antara objek satu dengan objek lainnya