BAB II
KAJIAN LITERATUR

2.1 Konsep Sistem Informasi

Konsep sistem informasi dalam penelitian ini mencakup tiga bagian utama, yaitu sistem informasi, akademik, dan sistem informasi akademik, yang saling berkaitan dan menjadi dasar dalam memahami proses pengelolaan data serta penerapannya dalam lingkungan pendidikan.

2.1.1 Sistem Informasi

Sistem adalah jaringan prosedur yang saling terhubung dan bekerja sama untuk melakukan aktivitas spesifik yang bertujuan mencapai tujuan tertentu. Informasi adalah data yang telah diolah menjadi bentuk yang lebih berguna bagi penerimanya untuk pengambilan keputusan, baik di masa sekarang maupun di masa mendatang. Sistem informasi adalah kombinasi berbagai komponen teknologi informasi yang berkolaborasi untuk menghasilkan informasi, memfasilitasi saluran komunikasi dalam suatu organisasi atau kelompok. Sistem informasi terdiri dari beberapa komponen yang saling terhubung untuk mencapai tujuan yang diinginkan. Sistem informasi merepresentasikan hubungan antara data dan metode, yang memanfaatkan perangkat keras dan perangkat lunak untuk menyampaikan informasi yang bermanfaat [7].

2.1.2 Akademik

Akademik berasal dari istilah "akademi" yang merujuk pada institusi pendidikan tinggi seperti universitas, institut, atau sekolah tinggi. Secara luas, akademik dapat diartikan sebagai segala sesuatu yang berhubungan dengan ilmu pengetahuan, pendidikan, dan riset yang bersifat ilmiah dan teoretis. Untuk memahami makna akademik dengan lebih mendalam, penting untuk membedakannya dari aspek non-akademik. Berikut adalah beberapa perbedaan utama antara keduanya [8]:

1. Fokus dan Tujuan

   Akademik berorientasi pada peningkatan pengetahuan serta keterampilan yang berkaitan dengan bidang tertentu. Tujuan utamanya adalah untuk memperdalam pemahaman secara teoritis dan praktis dalam disiplin tertentu. Di sisi lain, non-akademik lebih

menekankan pada pengembangan keterampilan hidup, bakat, dan minat di luar kurikulum yang resmi.

2. Metode Penilaian

   Pencapaian akademik biasanya dievaluasi melalui ujian tulisan, tugas, penelitian, dan berbagai metode penilaian resmi lainnya. Sementara itu, pencapaian non-akademik sering kali diukur melalui kompetisi, penampilan, atau keberhasilan dalam kegiatan di luar kurikulum.

3. Sifat Kegiatan

   Kegiatan akademik umumnya bersifat terorganisir dan mengikuti kurikulum yang telah ditentukan. Kegiatan non-akademik cenderung lebih fleksibel dan dapat disesuaikan dengan minat serta bakat individu.

4. Pengakuan dan Sertifikasi

   Pencapaian akademik biasanya diberikan pengakuan melalui gelar, sertifikat, atau ijazah yang diterbitkan oleh lembaga pendidikan. Sementara prestasi non-akademik mungkin diakui melalui penghargaan, medali, atau bentuk pengakuan tidak resmi lainnya.

5. Aplikasi dalam Kehidupan

   Pengetahuan dan keterampilan yang diperoleh dari pendidikan akademik biasanya diterapkan dalam konteks profesional atau penelitian. Keterampilan non-akademik seringkali memiliki penggunaan yang lebih luas dalam kehidupan sehari-hari serta pengembangan diri.

2.1.3 Sistem Informasi Akademik

Sistem Informasi Akademik merupakan sebuah aplikasi yang mengelola semua aktivitas yang berhubungan dengan pendidikan di sekolah, termasuk informasi tentang siswa, guru, jadwal kelas, dan nilai, dengan cara yang sistematik dan mudah diakses. Dengan adanya sistem ini, diharapkan setiap langkah dalam lingkungan sekolah, terutama dalam pengelolaan data akademik, dapat berlangsung dengan lebih efisien, cepat, dan hasil yang tepat. Selain itu, penggunaan sistem ini juga memberikan kemudahan bagi pihak sekolah dalam mengorganisir berbagai kegiatan, sehingga layanan pendidikan menjadi lebih terstruktur dan hasil kerja menjadi optimal. Secara umum, proses bisnis yang berjalan sebelum adanya sistem informasi akademik seperti [9] :

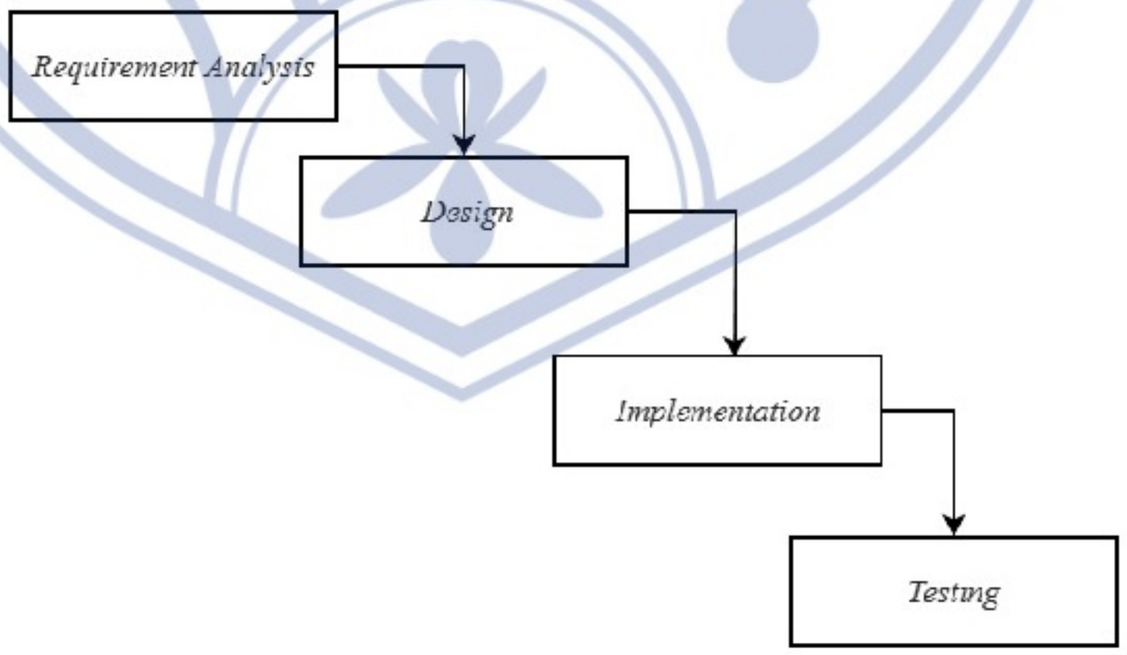
1. Pengolahan data siswa
2. Pengolahan data guru
3. Pembuatan jadwal Pelajaran

4. Pembagian kelas
5. Proses penilaian siswa
6. Pemeriksaan nilai oleh wali kelas
7. Penyampaian nilai kepada siswa
8. Pencarian data akademik
9. Penyampaian informasi akademik

Sistem informasi akademik sekolah dapat membantu mengelola data sekolah dengan cepat dan mudah. Selain mempermudah pengolahan data yang sebelumnya dilakukan secara manual, sistem informasi akademik juga dapat digunakan sebagai cara untuk menyimpan data dengan aman [10].

2.2 Metode *Waterfall*

Metode *waterfall* adalah suatu cara dalam pengembangan perangkat lunak yang bersifat linear dan berurutan. Para pengembang memanfaatkan metode ini sebagai salah satu teknik awal dalam pengelolaan proyek perangkat lunak. Istilah *Waterfall* diambil dari karakteristik alur kerjanya yang mengalir satu arah, mirip seperti aliran air terjun, dari satu tahap ke tahap berikutnya tanpa kembali ke fase sebelumnya. Model ini memiliki struktur yang kaku dengan tahapan yang harus diselesaikan secara berurutan. Setiap tahap harus diselesaikan sepenuhnya sebelum melanjutkan ke tahap berikutnya. Metode ini dikenal sebagai bagian dari proses *Software Development Life Cycle* (SDLC) yang bersifat tradisional [11].



Gambar 2. 1 Tahapan dalam Metode *Waterfall*

Tahapan dalam metode *Waterfall* terdiri dari *Requirement Analysis, System Design, Implementation,* dan *Testing* yang dapat dilihat pada Gambar 2.1 [12].

1. *Requirement Analysis*

   Pada fase ini, pengembang sistem harus melakukan komunikasi untuk memahami perangkat lunak yang diinginkan oleh pengguna dan batasan yang ada. Informasi ini sering kali dikumpulkan melalui wawancara, survei, atau diskusi.

2. *System Design*

   Dalam proses desain, dilakukan penerjemahan kebutuhan ke dalam sebuah rencana desain perangkat lunak yang dapat diprediksi sebelum proses pengkodean dimulai. Fokus pada tahap ini adalah pada struktur data, arsitektur perangkat lunak, representasi antarmuka, dan rincian algoritma prosedural.

3. *Implementation*

   Pada tahap ini, rancangan desain diterjemahkan menjadi bentuk yang bisa dipahami oleh mesin, dengan menggunakan kode dari bahasa pemrograman. Kode yang dihasilkan masih berupa modul-modul kecil yang akan digabungkan pada tahap berikutnya.

4. *Testing*

   Di tahap ini, modul-modul yang telah dibuat digabungkan dan dilakukan pengujian untuk memastikan apakah perangkat lunak yang dikembangkan sesuai dengan desain serta mengecek adanya kesalahan dalam fungsionalitas perangkat lunak.

2.3 *Website*

Website merupakan sekumpulan halaman situs yang terletak dalam suatu domain atau sub domain yang ada di *World Wide Web* (WWW) di internet. *Website* terdiri dari *Page* atau halaman dan sekumpulan halaman yang disebut *Homepage*. *Homepage* terletak di posisi paling atas diikuti oleh halaman-halaman lain yang berada di bawahnya. Umumnya, setiap halaman yang berada di bawah *Homepage* (*Child Page*) memiliki *Hyper-Link* yang mengarah ke halaman lain di dalam *Website* [13]. Secara umum, *Website* dibagi menjadi tiga jenis [14]:

1. *Website* Statis

   Website statis yaitu jenis *Website* yang isinya tidak diperbaharui secara berkala, sehingga isinya dari waktu ke waktu akan selalu tetap. *Website* jenis ini biasanya hanya digunakan untuk menampilkan profil dari pemilik *Website* seperti profil perusahaan atau organisasi. Untuk saat ini, *Website* jenis ini banyak digunakan pada website jenis *Landing Page*.

2. *Website* Dinamis

   Website dinamis yaitu jenis *Website* yang isinya terus diperbaharui secara berkala oleh pengelola web atau pemilik *Website*. *Website* jenis ini banyak dimiliki oleh perusahaan atau perorangan yang aktivitas bisnisnya memang berkaitan dengan internet. Contoh paling mudah dari website jenis ini yaitu *Web Blog* dan *Website* berita.

3. *Website* Interaktif

   Website interaktif pada dasarnya termasuk dalam kategori *Website* dinamis, di mana isi informasinya selalu diperbaharui dari waktu ke waktu. Hanya saja, isi informasi tidak hanya diubah oleh pengelola *Website* tetapi lebih banyak dilakukan oleh pengguna *Website* itu sendiri. Contoh *Website* jenis ini yaitu website jejaring sosial seperti *Facebook* dan *Twitter* atau *Website Marketplace* seperti *Bukalapak, Tokopedia, Shopee,* dan sebagainya.

2.4 *Xampp*

XAMPP merupakan perangkat lunak server web yang siap digunakan, dapat dioperasikan di sistem *Linux* maupun *Windows*. Tujuan utamanya adalah berfungsi sebagai server mandiri (*Localhost*), yang meliputi program *Apache* HTTP server, basis data MySQL, serta interpreter untuk bahasa yang ditulis dalam PHP dan Perl. Istilah XAMPP adalah akronim dari X (empat jenis sistem operasi), *Apache*, MySQL, PHP, dan Perl. Perangkat lunak ini dirilis di bawah GNU *General Public License* dan bersifat gratis, menawarkan kemudahan dalam penggunaan serta kemampuan untuk menyajikan halaman web yang bersifat dinamis [15].

2.5 *Framework Laravel*

Laravel adalah *framework PHP* yang bersifat *Open Source*, dirilis dengan lisensi MIT dan dibangun berdasarkan konsep MVC (*Model View Controller*). Ini adalah alat untuk membuat situs web dengan pendekatan MVC yang ditulis menggunakan PHP, yang ditujukan untuk meningkatkan kualitas perangkat lunak sambil mengurangi biaya perawatan. *Laravel* adalah salah satu *framework PHP* terbaik yang diciptakan oleh *Taylor Otwell*. Sebagai *framework PHP*, *Laravel* berfungsi sebagai *Platform* pengembangan web yang bersifat *Open Source*. Gaya ekspresi dan sintaks dalam *Laravel* juga sangat menarik. Ini dirancang khusus untuk mempermudah dan mempercepat proses pengembangan web [16].

Beberapa fitur yang ada di *Laravel* adalah sebagai berikut [17]:

1. *Bundles*, yaitu fitur yang menggunakan sistem pengemasan modular dan menyediakan berbagai pilihan di aplikasi.
2. *Eloquent ORM* adalah implementasi *PHP* lanjutan yang menawarkan metode internal berdasarkan pola "*Active Record*" untuk menyelesaikan masalah yang berkaitan dengan hubungan objek di *Database*.
3. *Application Logic* adalah komponen dalam aplikasi yang menggunakan *Controller* atau bagian dari *Route*.
4. *Reverse Routing* menjelaskan relasi antara *Link* dan *Route*.
5. *Restful controllers* memisahkan logika dalam menangani permintaan *HTTP GET* dan *POST*.
6. *Class Auto Loading* memberikan kemudahan dalam memuat *Class PHP* secara otomatis.
7. *View Composer* adalah kode logis yang dapat dijalankan ketika view sedang *Loading*.
8. *IoC Container* memungkinkan pembuatan objek baru dengan membalik *Controller*.
9. *Migration* menyediakan sistem kontrol untuk struktur *Database*.
10. *Unit Testing* menyediakan berbagai tes untuk mendeteksi dan menghindari regresi.
11. *Automatic Pagination* menyederhanakan proses dalam penerapan halaman.

2.6 Basis Data

Basis data adalah sekumpulan informasi yang saling berhubungan yang disimpan bersama pada suatu tempat, diatur berdasarkan suatu skema atau struktur tertentu, dan dilengkapi dengan perangkat lunak untuk melakukan pengolahan untuk tujuan tertentu. Basis data terdiri dari sekumpulan data yang saling terhubung dan tersimpan dalam perangkat keras (*Hardware*) komputer, sementara perangkat lunak (*Software*) komputer digunakan untuk melakukan berbagai pengolahan data, seperti pembaruan, pencarian, dan mengolahnya dengan perhitungan [18]. Adapun tujuan dan manfaat basis data di antaranya [19]:

1. Kecepatan dan Kemudahan

   Memungkinkan untuk melakukan perubahan/manipulasi terhadap data atau menampilkan kembali data dengan lebih cepat dan mudah.
2. Efisiensi Ruang Penyimpanan

   Efisiensi/ optimalisasi penggunaan ruang penyimpanan dengan melakukan penekanan (menghilangkan) redundansi data.

3. Keakuratan

   Menerapkan aturan/ batasan (*Constraint*) tipe data, domain data, atau keunikan data untuk menghindari pemasukan data yang tidak akurat.

4. Ketersediaan

   Memilah data menjadi data master tersimpan dalam suatu server, data transaksi ataupun data *History* tersedia dengan kapasitas yang besar.

5. Kelengkapan

   Menambah *Record-Record* data dan melakukan perubahan struktur dalam basis data baik dalam bentuk penambahan objek baru (tabel) atau dengan penambahan *Field-Field* baru pada tabel.

6. Keamanan

   Melakukan pengaturan hak akses terhadap basis data beserta objek-objek di dalamnya dan menentukan operasi-operasi apa saja yang boleh dilakukan.

7. Kebersamaan Pemakai

   Penggunaan data dalam suatu basis data oleh berbagai pihak. Data yang terdapat dalam database dapat digunakan secara bersamaan kapan dan dimana saja, tersimpan secara *Cloud*. Membantu mempercepat pekerjaan, bisa kolaborasi dalam sistem pendataan, bisa menggunakan sistem terdistribusi yang dapat menerima, mengirim dan menggunakan kembali data-data dalam suatu *Database System*.

2.7 MySql

MySQL merupakan salah satu jenis server basis data yang populer, yang tergolong dalam kategori RDBMS (*Relational Database Management System*). *MySQL* adalah server RDBMS (*Relational Database Management System*). RDBMS adalah perangkat lunak yang memungkinkan pengguna basis data untuk merancang, mengelola, dan memproses informasi dalam suatu model relasional. Oleh karena itu, tabel-tabel yang terdapat dalam basis data memiliki hubungan satu sama lain [20].

Keunggulan dan Kelebihan DBMS MySQL [21]:

1. Merupakan salah satu perangkat lunak yang portabel

   MySQL memiliki kelebihan utama, yaitu termasuk dalam kategori perangkat lunak yang portabel. Artinya, MySQL dapat digunakan untuk mengelola *Database* di berbagai *Platform*.

2. MySQL adalah DBMS yang bersifat *Opensource*

   Salah satu keunggulan terbesar dari MySQL adalah bahwa perangkat ini tersedia secara gratis. Ya, versi dasar MySQL dapat diakses tanpa biaya karena tergolong perangkat lunak *Open Source.*

3. *Multi-User*

   Seperti halnya DBMS lainnya, meskipun bersifat *Open Source*, MySQL memiliki kemampuan yang sangat baik dalam mendukung penggunaan oleh banyak pengguna sekaligus. Banyak pengguna bisa mengaksesnya pada waktu yang sama tanpa mengalami masalah seperti kerusakan sistem atau kendala lainnya.

4. Memiliki berbagai jenis data

   Jenis data yang disediakan oleh MySQL sangat beragam. Beberapa jenis data yang ada dalam MySQL meliputi *Integer, Float, Double, Char, Text, Date, Timestamp*, dan masih banyak lagi.

5. Memiliki fitur keamanan yang handal

   Kelebihan lain dari MySQL adalah fitur keamanannya yang cukup handal, terutama mengingat bahwa perangkat lunak ini bersifat *Open Source*, yang berarti dapat digunakan tanpa biaya, fitur keamanan yang ada sudah sangat efektif.

6. Alat administrasi yang komprehensif

   Alat-alat administrasi yang terdapat dalam perangkat lunak ini juga sudah cukup *Komprehensif*. Pengguna dan pengembang dapat menggunakan MySQL dengan mudah, tanpa perlu belajar secara mendalam tentang MySQL.

2.8 *Black Box Testing*

Black box testing adalah teknik dalam pengujian perangkat lunak yang tidak memerlukan pengetahuan tentang struktur internal atau kode dari aplikasi yang sedang diuji. Metode ini sering disebut juga sebagai pengujian fungsional atau pengujian berbasis input. Ada enam teknik utama yang diterapkan dalam pengujian ini, yaitu *Equivalence Class Partitioning, Boundary Value Analysis, Cause Effect Graphing. Fuzzy Testing,* dan model *Based Testing*. Dengan menerapkan teknik-teknik ini, *Black box testing* dapat meningkatkan keandalan serta kualitas perangkat lunak dari perspektif pengguna [22]. Kelebihan dari metode *Black Box Testing* adalah tidak memerlukan pengetahuan tentang bahasa pemrograman yang digunakan. Pengujian ini dilakukan berdasarkan sudut pandang pengguna aplikasi, sehingga dapat dengan segera mengidentifikasi kekurangan yang perlu

diperbaiki pada aplikasi, seperti ketidakjelasan dan inkonsistensi dalam spesifikasi fungsionalnya [23].

2.9 Teknik Pengembangan Sistem

Teknik Pengembangan Sistem Informasi digunakan untuk membantu menganalisis kebutuhan, menggambarkan proses kerja, serta mengidentifikasi masalah dalam sistem yang akan dibangun. Melalui metode seperti *Use Case* Diagram, *Activity* Diagram, Analisis PIECES, *Flowchart,* dan *Fishbone* Diagram, pengembang dapat memahami alur sistem secara menyeluruh sehingga proses perancangan dapat dilakukan dengan lebih terstruktur dan efektif.

2.9.1 *Use Case* Diagram

Use Case Diagram merupakan salah satu jenis diagram dalam *Unified Modeling Language* (UML) yang digunakan untuk menggambarkan bagaimana suatu sistem digunakan serta menunjukkan hubungan antara actor dan system. Analis biasanya memulai perancangan dengan membuat diagram kasus penggunaan ini agar dapat memahami kebutuhan dan ruang lingkup sistem secara lebih jelas. Melalui diagram ini, pengembang dapat mengetahui interaksi yang terjadi antara pengguna dan sistem, sehingga proses pengembangan dapat berjalan lebih terarah, terstruktur, dan sesuai dengan kebutuhan yang telah ditentukan [24]. Komponen-komponen pada *Use Case* diagram [25]:

1. Aktor

   Aktor merupakan komponen yang bertugas untuk menunjukkan siapa yang berhubungan dengan sistem. komponen ini berfungsi sebagai alat yang mengirimkan dan menerima data dari sistem. Kedua fungsi ini dapat berlangsung bersamaan.

2. Sistem

   Sistem dalam diagram *Use Case* untuk penjualan daring dan hal-hal lainnya digambarkan dengan persegi. Biasanya, sistem tidak disertakan dengan gambar, karena itu bukan cara yang tepat untuk memahaminya. Sistem akan diidentifikasi dengan label yang sesuai. Fungsi dari sistem itu sendiri adalah untuk mendefinisikan batasan *Use Case* melalui interaksi dari luar sistem.

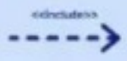
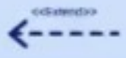
3. *Use Case*

   Komponen yang dikenal sebagai *Use Case* merupakan bagian yang menggambarkan fungsi dalam sistem informasi. Dengan demikian, pengembang dan pengguna dapat dengan mudah memahami cara kerja sistem yang direncanakan.

4. *Relationship*

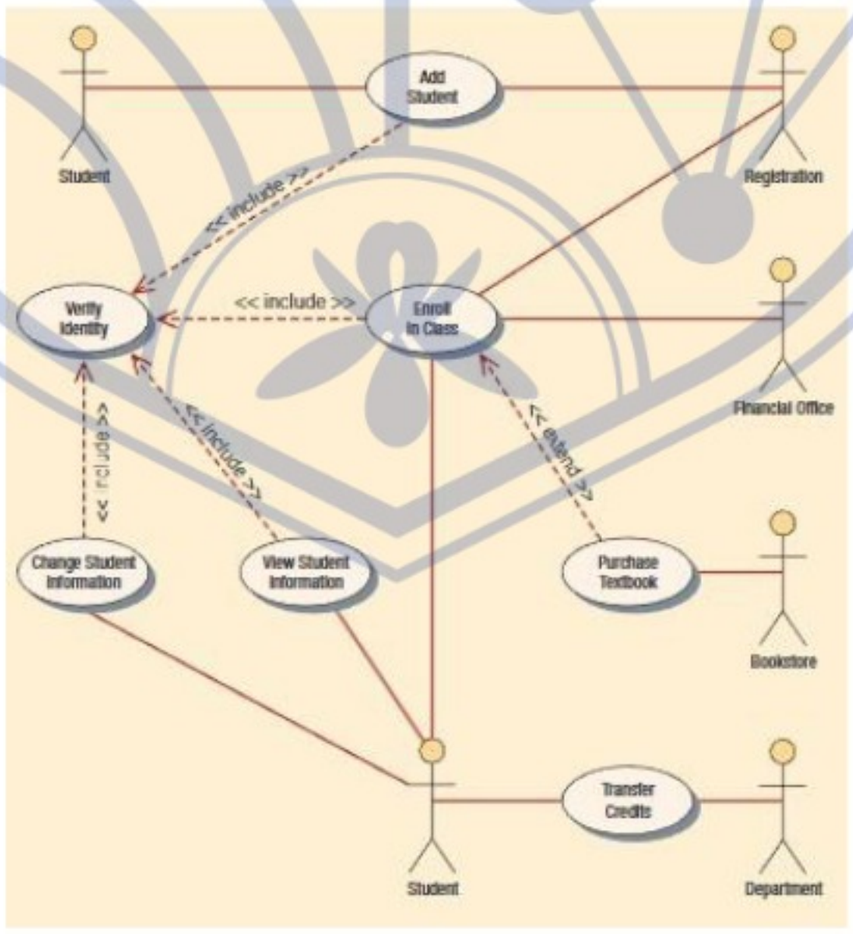
Relationship atau hubungan merupakan simbol *Use Case* yang menggambarkan relasi antara *Use Case* dengan aktor. Di dalam sistem informasi, hubungan ini dapat dibagi menjadi beberapa kategori, seperti *Generalization, Association, Include, Dan Extend.*

Simbol- Simbol pada *Use Case Diagram* yaitu [26]:

Simbol	Nama	Keterangan
	Actor	Entitas yang berinteraksi dengan sistem
	User case	Aktivitas yang dapat dilakukan actor pada sistem
	Association	Hubungan antara actor dengan use case
	System	Sistem yang sedang dikembangkan
	Include	Suatu use case termasuk bagian dari use case lain
	Extend	Satu use case dapat diperluas dengan use case lain
	Generalization	Satu actor atau use case merupakan generalisasi dari yang lain

Gambar 2. 2 Simbol *Use Case* Diagram

Contoh pada *Use Case* Diagram[24].



Gambar 2. 3 Contoh *Use Case* Diagram

2.9.2 *Use Case Scenario*

Use Case Scenario menggambarkan aktor yang terlibat dalam proses dalam sistem, serta menjelaskan bagaimana sistem memberikan tanggapan terhadap tindakan yang diambil oleh aktor tersebut. berfungsi untuk menggambarkan interaksi yang berlangsung antara pengguna dan sistem. *Use Case Scenario* berfungsi untuk menggambarkan interaksi yang berlangsung antara pengguna dan system [27]. Contoh dari *Use Case Scenario*[24].

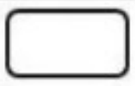
Use case name:	Change Student Information	**UniqueID:** Student UC 005
Area:	Student System	
Actor(s):	Student	
Description:	Allow student to change his or her own information, such as name, home address, home telephone, campus address, campus telephone, cell phone, and other information using a secure website.	
Triggering Event:	Student uses Change Student Information website, enters student ID and password, and clicks the **Submit** button.	
Trigger type:	☒ External ☐ Temporal	
Steps Performed (Main Path)		**Information for Steps**
1. Student logs on to the secure web server.		Student ID, Password
2. Student record is read and password is verified.		Student Record, Student ID, Password
3. Current student personal information is displayed on the Change Student web page.		Student Record
4. Student enters changes on the Change Student Web form and clicks **Submit** button.		Change Student Web Form
5. Changes are validated on the web server.		Change Student Web Form
6. Change Student Journal record is written.		Change Student Web Form
7. Student record is updated on the Student Master.		Change Student Web Form, Student Record
8. Confirmation web page is sent to the student.		Confirmation Page
Preconditions:	Student is on the Change Student Information web page.	
Postconditions:	Student has successfully changed personal information.	
Assumptions:	Student has a browser and a valid user ID and password.	
Requirements Met:	Allow students to be able to change personal information using a secure website.	
Outstanding Issues:	Should the number of times a student is allowed to logon be controlled?	
Priority:	Medium	
Risk:	Medium	

Gambar 2. 4 Contoh *Use Case Scenario*

2.9.3 *Activity* Diagram

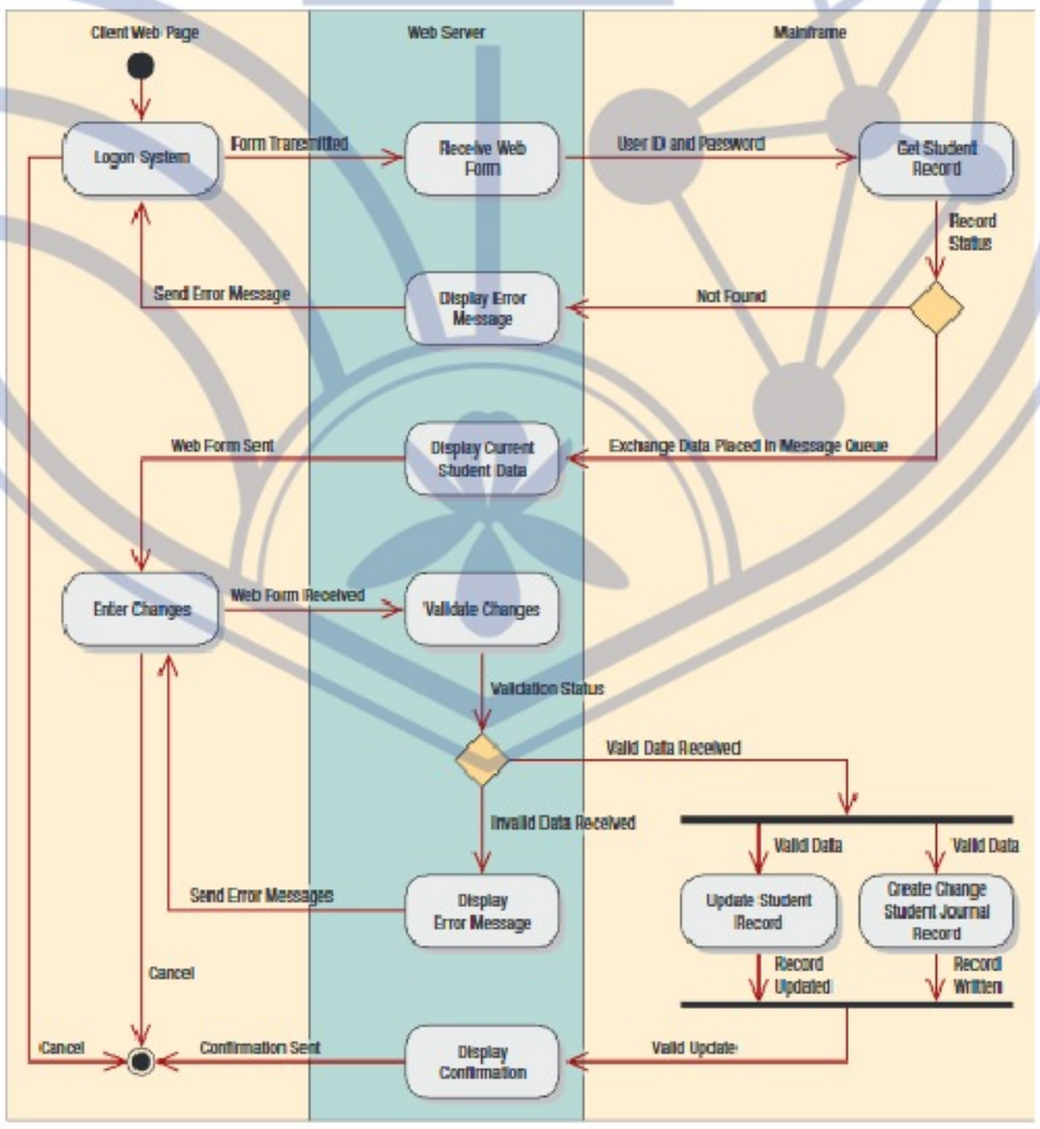
Activity diagram merupakan salah satu jenis diagram dalam UML (*Unified Modeling Language*) yang dipakai untuk menunjukkan alur kerja atau proses dalam perangkat lunak atau bisnis. Diagram ini biasanya digunakan saat menganalisis dan merancang perangkat lunak. Dengan diagram ini, pengguna dapat menggambarkan berbagai aktivitas, aliran kontrol(*Control Flow*), serta hubungan antara aktivitas dalam sebuah sistem. Di dalamnya terdapat berbagai komponen dengan bentuk tertentu yang saling terhubung melalui panah. Panah tersebut menunjukkan urutan aktivitas yang berlangsung dari awal sampai akhir proses. *Activity* diagram sering kali disebut sebagai '*behaviour* diagram' karena dapat menampilkan berbagai aspek dinamis dari sistem yang sedang dimodelkan. Diagram aktivitas menampilkan langkah-langkah dalam sebuah proses, seperti tindakan yang dilakukan berurutan atau bersamaan, serta pilihan yang diambil. Diagram aktivitas biasanya dibuat untuk satu kasus penggunaan dan bisa menampilkan beberapa skenario yang mungkin

terjadi[24]. Komponen yang ada pada *Activity* diagram dapat dilihat pada gambar dibawah ini [28].

Simbol	Nama	Keterangan
	Status awal	Sebuah diagram aktivitas memiliki sebuah status awal.
	Aktivitas	Aktivitas yang dilakukan sistem,aktivitas biasanya diawali dengan kata kerja.
	Percabangan / Decision	Percabangan dimana ada pilihan aktivitas yang lebih dari satu.
	Penggabungan / Join	Penggabungan dimana yang mana lebih dari satu aktivitas lalu digabungkan jadi satu.
	Status Akhir	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
	Swimlane	Swimlane memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

Gambar 2. 5 Komponen pada *Activity Diagram*

Contoh dari *Activity* Diagram [24].



Gambar 2. 6 Contoh *Activity* Diagram

2.9.4 Analisis PIECES

Analisis PIECES (*Performance, Information, Economy, Control, Eficiency, dan Service*) adalah metode untuk menemukan dan menyelesaikan masalah yang muncul baik bagi pelanggan maupun bagi pihak internal perusahaan, guna mengevaluasi sistem yang berjalan dan sistem usulan [29]. Untuk mengevaluasi sebuah sistem informasi dengan metode PIECES, terdapat enam faktor penilaian yang perlu diperhatikan:

1. *Performance* (Kinerja)

   Untuk menilai seberapa baik sistem informasi dapat berfungsi sesuai dengan tujuan pengembangannya.

2. *Information* (Informasi)

   Untuk memeriksa seberapa tepat kualitas informasi yang dihasilkan oleh sistem, mulai dari data masuk hingga keluaran.

3. *Economic* (Ekonomi)

   Yang menganalisis pengeluaran yang dikeluarkan untuk operasional sistem serta nilai tambah yang diperoleh selama sistem digunakan.

4. *Control* (Pengendalian)

   Yaitu mengukur seberapa baik sistem informasi dalam mendeteksi kesalahan atau kecurangan yang mungkin terjadi.

5. *Eficiency* (Efisiensi)

   Yang menilai seberapa efisien sistem informasi ini selama proses berlangsung.

6. *Service*(Layanan)

   Yang menilai aspek pelayanan dalam meningkatkan kualitas sistem secara keseluruhan untuk kepuasan pengguna.

2.9.5 *Flowchart*

Flowchart merupakan gambaran visual dari sebuah proses atau tahapan kerja yang disusun dengan cara yang teratur menggunakan simbol-simbol tertentu. *Flowchart* bertujuan untuk menerangkan langkah-langkah atau rangkaian kegiatan dalam menyelesaikan sebuah tugas atau permasalahan.

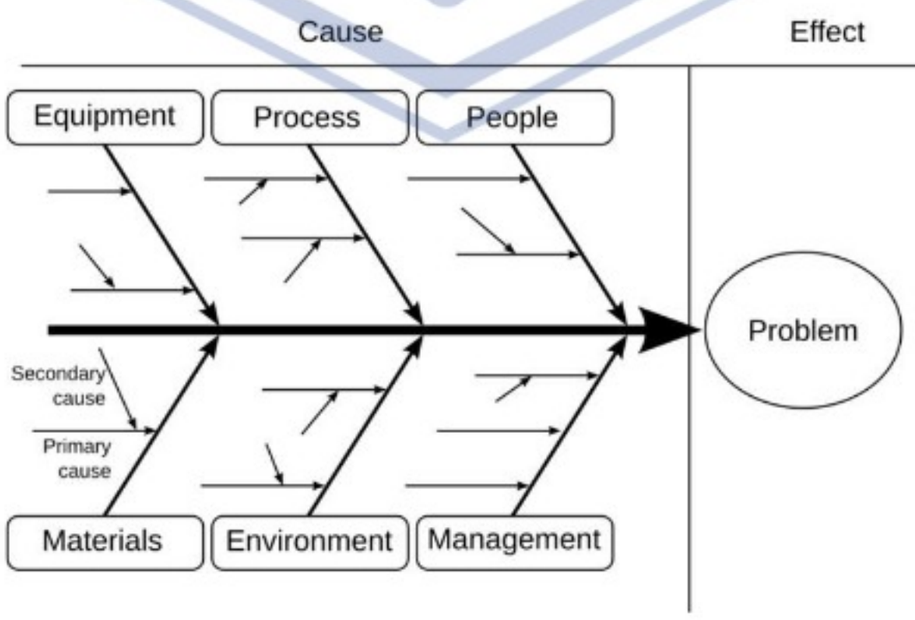
Flowchart memiliki beberapa simbol yaitu [30]:

→ ↑ ↓ ←	**Flow** Simbol yang digunakan untuk menggabungkan antara simbol yang satu dengan simbol yang lain. Simbol ini disebut juga dengan Connecting Line.	▱	**Input/output** Simbol yang menyatakan proses input atau output tanpa tergantung peralatan.
○	**On-Page Reference** Simbol untuk keluar - masuk atau penyambungan proses dalam lembar kerja yang sama.	⏢	**Manual Operation** Simbol yang menyatakan suatu proses yang tidak dilakukan oleh komputer.
⬠	**Off-Page Reference** Simbol untuk keluar - masuk atau penyambungan proses dalam lembar kerja yang berbeda.	▭~	**Document** Simbol yang menyatakan bahwa input berasal dari dokumen dalam bentuk fisik, atau output yang perlu dicetak.
⬭	**Terminator** Simbol yang menyatakan awal atau akhir suatu program.	▯▯	**Predefine Proses** Simbol untuk pelaksanaan suatu bagian (sub-program) atau prosedure.
▭	**Process** Simbol yang menyatakan suatu proses yang dilakukan komputer.	⌓	**Display** Simbol yang menyatakan peralatan output yang digunakan.
◇	**Decision** Simbol yang menunjukan kondisi tertentu yang akan menghasilkan dua kemungkinan jawaban, yaitu ya dan tidak.	⬡	**Preparation** Simbol yang menyatakan penyediaan tempat penyimpanan suatu pengolahan untuk memberikan nilai awal.

Gambar 2. 7 Simbol Dari *Flowchart*

2.9.6 *Fishbone* Diagram

Fishbone Diagram (Diagram Tulang Ikan) atau dikenal juga sebagai Diagram Sebab Akibat (*Cause and Effect Diagram*) adalah diagram yang menunjukkan hubungan antara sebab dan akibat [31]. Menggambarkan masalah dalam suatu diagram atau gambar memiliki tujuan untuk membuat kita lebih mudah memahami gambaran masalah dan faktor-faktor yang menyebabkan masalah muncul dalam diagram atau gambar tersebut. Konsep dasar dari diagram tulang ikan adalah bahwa masalah utama ada di bagian kanan diagram atau di kepala kerangka tulang ikan.



Gambar 2. 8 *Fishbone* Diagram

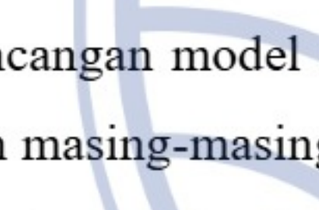
Langkah-langkah dalam penyusunan *Fishbone* Diagram atau CED yaitu [32]:

1. Tetapkan masalah yang akan diselesaikan atau ditangani.
2. Tuliskan masalah di bagian kanan dan buat panah kiri ke kanan.
3. Tuliskan faktor-faktor utama yang mempengaruhi atau mempengaruhi cabang utama. 4M (*Material, Method, Mechanism, dan Manpower*) atau 4P (*Parts (raw material), Procedures, Plant (equipment), dan People*) dapat digunakan untuk menentukan faktor-faktor utama masalah. Namun, tergantung pada masalahnya, kategori juga bisa berbeda.
4. Untuk masing-masing kelompok penyebab masalah, temukan penyebabnya dan tuliskan pada ranting berdasarkan kelompok faktor penyebab utama. Faktor-faktor yang menyebabkan masalah ini diteliti lebih lanjut dengan memeriksa faktor-faktor yang telah diidentifikasi sebelumnya dengan lebih rinci. Metode "5-*Whys*" dalam wawancara dan FGD dapat digunakan untuk mengetahui penyebab detail ini.
5. Pastikan diagram menunjukkan setiap detail penyebab masalah.

2.9.7 *Class* Diagram

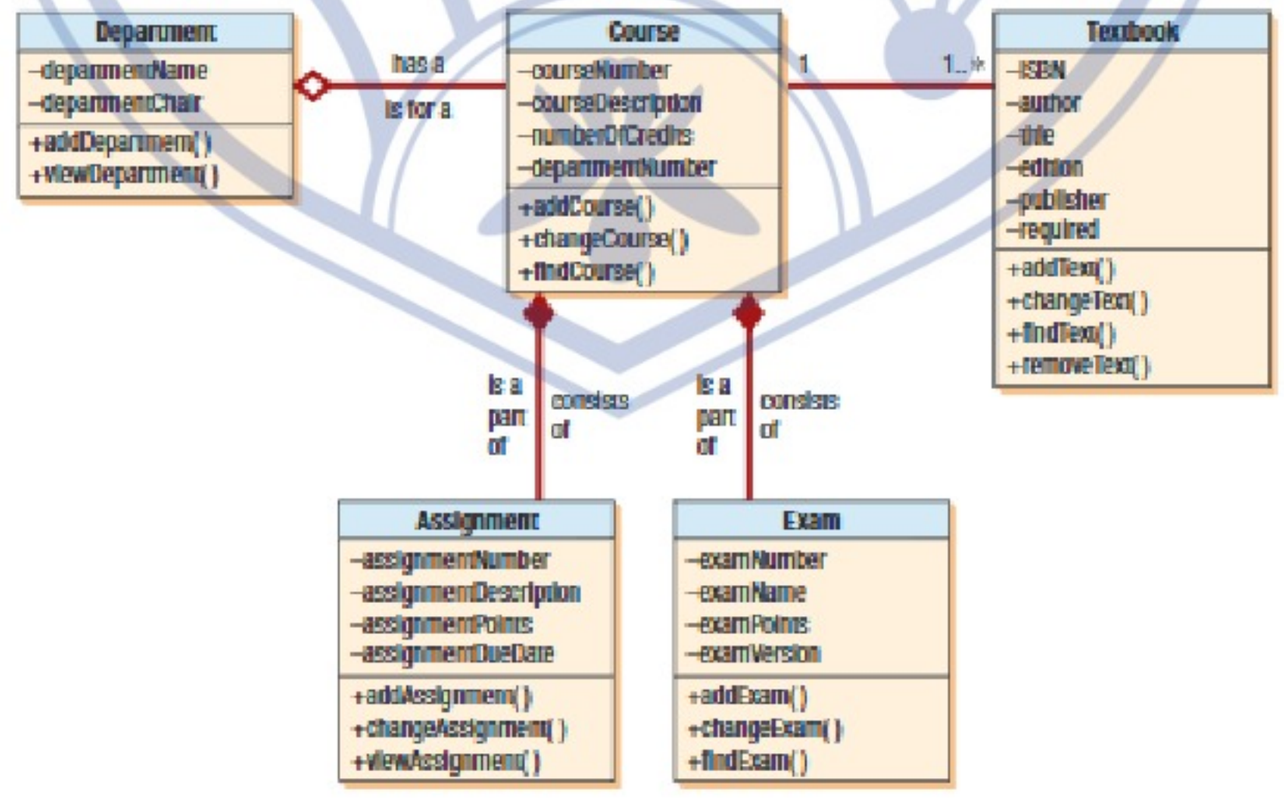
Class Diagram bisa hanya menunjukkan nama kelas saja; atau nama kelas beserta atributnya; atau nama kelas, atribut, dan metode juga [24]. *Class* diagram merupakan gambaran tentang keterkaitan antar kelas serta penjelasan mendetail mengenai setiap kelas dalam rancangan model suatu sistem. Diagram ini juga menunjukkan prinsip-prinsip dan kewajiban masing-masing entitas yang memengaruhi perilaku sistem. *Class* diagram sering kali disebut sebagai sekumpulan beberapa kelas beserta relasinya. *Class* itu sendiri serupa dengan entitas yang diwakili dalam bentuk persegi, di mana nama kelas dicantumkan di bagian atas, kemudian di bawahnya terdapat atribut-atribut yang dimiliki *Class* tersebut, dan di bawahnya lagi tertera metode yang ada di dalam *Class*. *Class* Diagram mempunyai simbol-simbol dalam penggunaannya pada Tabel 2. 1 [33].

Tabel 2. 1 Simbol *Class* Diagram

Simbol	Nama	Keterangan
	*Class*	*Class* adalah blok-blok Pembangunan pada pemrograman berorentasi objek.
	Association	*Association* adalah sebuah hubungan yang

		menunjukan adanya interaksi antar *Class*.
	*Aggregation*	*Aggregation* mengindikasikan keseluruhan bagian *Relationship* dan biasanya disebut relasi
	*Composition*	Jika sebuah *Class* tidak bisa berdiri sendiri dan harus merupakan bagian dari *Class* yang lain, maka class tersebut memiliki relasi *Composition* terhadap class tempat dia bergantung tersebut.
	Generalization	*Generalization* adalah sebuah hubungan antar class yang bersifat dari khusus ke umum.
	*Dependency*	Umumnya penggunaan *Dependency* digunakan untuk menunjukkan operasi pada suatu class yang menggunakan *Class* yang lain.

Contoh *Class Diagram* [24].



Gambar 2. 9 Contoh *Class* Diagram