BAB II
KAJIAN LITERATUR

2.1 Konsep Sistem Informasi

Sistem merupakan kumpulan dari elemen yang saling berkaitan di mana setiap elemen melakukan kegiatan bersama untuk mencapai tujuan tertentu. Sistem mempunyai model umum, yaitu *input*, proses, dan *output*[4]. Informasi merupakan data yang telah diproses sehingga meningkatkan pengetahuan dan memberi manfaat bagi seseorang yang menggunakan data tersebut. Informasi memiliki peran yang sangat penting dalam proses pengambilan keputusan, karena dapat memberikan data yang jelas, akurat, dan valid untuk mendukung keputusan yang tepat[5].

Sistem informasi merupakan seperangkat komponen yang bertujuan untuk mengumpulkan, memproses, dan menyimpan data mentah yang saling berkaitan menjadi informasi yang berguna untuk pengguna sebagai pengambilan keputusan . Selain itu, sistem informasi juga mendukung koordinasi, analisis, dan kontrol dalam suatu organisasi[6].

Komponten utama sistem informasi adalah elemen-elemen yang saling berinteraksi untuk mengumpulkan, memproses, menyimpan, dan menyebarkan informasi. Berikut komponen utama dari sistem informasi, yaitu[6]:

1. Perangkat Keras (*Hardware*)

   Perangkat keras merupakan komponen seperti komputer, server, dan perangkat *input* atau *output* lainnya. Fungsi *hardware* adalah mendukung proses pengolahan data menjadi informasi dan berfungsi sebagai media untuk menjalankan perangkat lunak.
2. Perangkat Lunak (*Software*)

   Perangkat lunak merupakan program yang digunakan untuk mengolah data menjadi informasi melalui *hardware*. Perangkat lunak terdiri dari perangkat lunak sistem (sistem operasi), perangkat lunak aplikasi (pengolah kata), dan perangkat lunak khusus.
3. Data

   Data merupakan informasi mentah berupa angka, teks, gambar, suara, dan video. Data berfungsi sebagai bahan mentah untuk menghasilkan informasi yang diolah melalui sistem informasi sehingga menghasilkan *output* yang bermanfaat dan relevan bagi pengguna.
4. Manusia (*Brainware*)

Manusia sebagai pengguna akhir (*end-user*), administrator, dan pengembang sistem yang berperan dalam mengoperasikan, mengelola, dan memanfaatkan sistem informasi yang mendukung operasional pengelolaan data.

5. Prosedur (*Procedure*)

   Prosedur merupakan langkah-langkah proses yang terkoordinasi dalam mengelola dan mengolah data. Proses bertujuan untuk memastikan data diolah dengan cara yang efisien dan akurat untuk menghasilkan informasi yang bermanfaat dan relevan bagi pengguna.

Gambar 2. 1 Komponen Sistem Informasi

2.2 *Object Oriented Analysis and Design* (OOAD)

Object Oriented Analysis and Design (OOAD) merupakan metode pendekatan dalam perancangan sistem informasi yang melakukan pendekatan pada objek dan dianggap sebagai pendekatan yang inovatif karena menekankan penggunaan model yang mempresentasikan konsep dunia nyata, menggunakan objek sebagai dasar, dan mengintegrasikan struktur data serta perilaku dalam entitas[7]. OOAD terdiri dari dua tahap utama, yaitu analisis beriorientasi objek (OOA) dan desain berorientasi objek (OOD). Analisis beriorientasi objek (OOA) bertujuan melakukan identifikasi kebutuhan pengguna dan mempresentasikan elemen-elemen dalam sistem sebagai objek, sedangkan desain berorientasi objek (OOD) bertujuan mengubah hasil OOA menjadi model yang diimplementasikan dalam sistem[8].

Metode *Object Oriented Analysis and Design* (OOAD) memiliki empat tahapan, yaitu sebagai berikut[9]:

1. *Planning*

Tahap *planning* melibatkan pengumpulan data melalui wawancara langsung dengan mitra untuk mendapatkan informasi yang dibutuhkan. Hasil dari wawancara tersebut adalah terbentuknya rumusan masalah untuk mengidentifikasi permasalahan utama yang dihadapi oleh mitra dan memberikan semua informasi yang dibutuhkan dalam proses pengembangan sistem.

2. *Analysis*

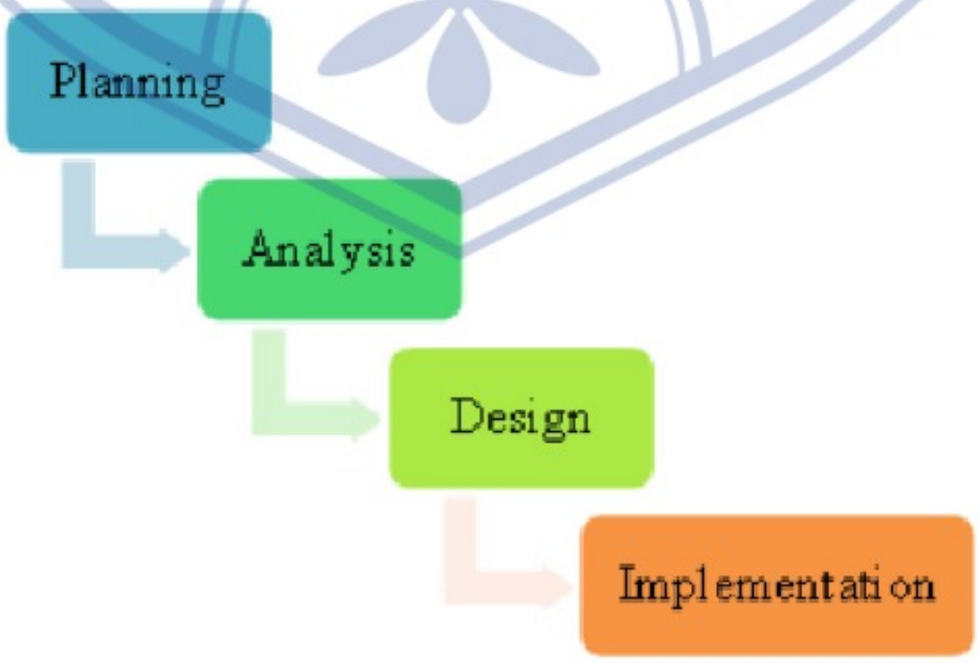
   Tahap *analysis* dilakukan untuk memahami kebutuhan mitra dan masalah yang akan dipecahkan oleh sistem sehingga dapat diketahui apa masalah bisnis dan target yang ingin dicapai. Tahap *analysis* akan dibantu dengan menggunakan diagram seperti *Use Case Diagram*, *Sequence Diagram*, *Activity Diagram*, dan *Class Diagram*. Masing-masing diagram akan membantu dalam merincikan konsep-konsep dan memberikan pondasi yang kuat untuk merancang sistem informasi yang efisien dan sesuai dengan kebutuhan mitra.

3. *Design*

   Tahap *design* dimulai dengan perancangan rinci dari sistem yang akan dibangun. Ini mencakup perancangan antarmuka pengguna. OOAD digunakan untuk memberikan arahan dan petunjuk dalam menciptakan sistem, memeriksa *requirements* dari sudut pandang objek dalam ruang lingkup permasalahan.

4. *Implementation*

   Tahap *implementation* merupakan tahapan akhir dari OOAD yang bertujuan untuk menyerahkan sistem yang sudah jadi dan melakukan uji coba. Pada tahap *implementation*, mitra memberikan *feedback* dan observasi terhadap sistem yang sudah dibuat.



Gambar 2. 2 Tahap *Object Oriented Analysis and Design* (OOAD)

2.3 Teknik Pengembangan Sistem

2.3.1 Model Use Case

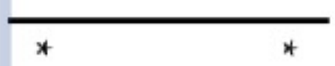
Model *Use Case* merupakan salah satu konsep utama dalam analisis dan perancangan sistem berbasis *object-oriented*. *Use case* menggambarkan bagaimana seorang pengguna (*actor*) berinteraksi dengan sistem untuk melakukan suatu kegiatan atau mencapai tujuan tertentu. Selain itu, *Use Case* juga berfungsi sebagai sarana komunikasi antara pengguna dan pengembang sistem. Komunikasi dilakukan dengan menekankan pada apa yang dilakukan sistem, bukan bagaimana sistem melakukannya. Dengan begitu, pengguna non-teknis pun dapat memahami rancangan sistem, memberikan masukan, atau memverifikasi apakah sistem yang dibuat sudah sesuai dengan kebutuhan mereka[10].

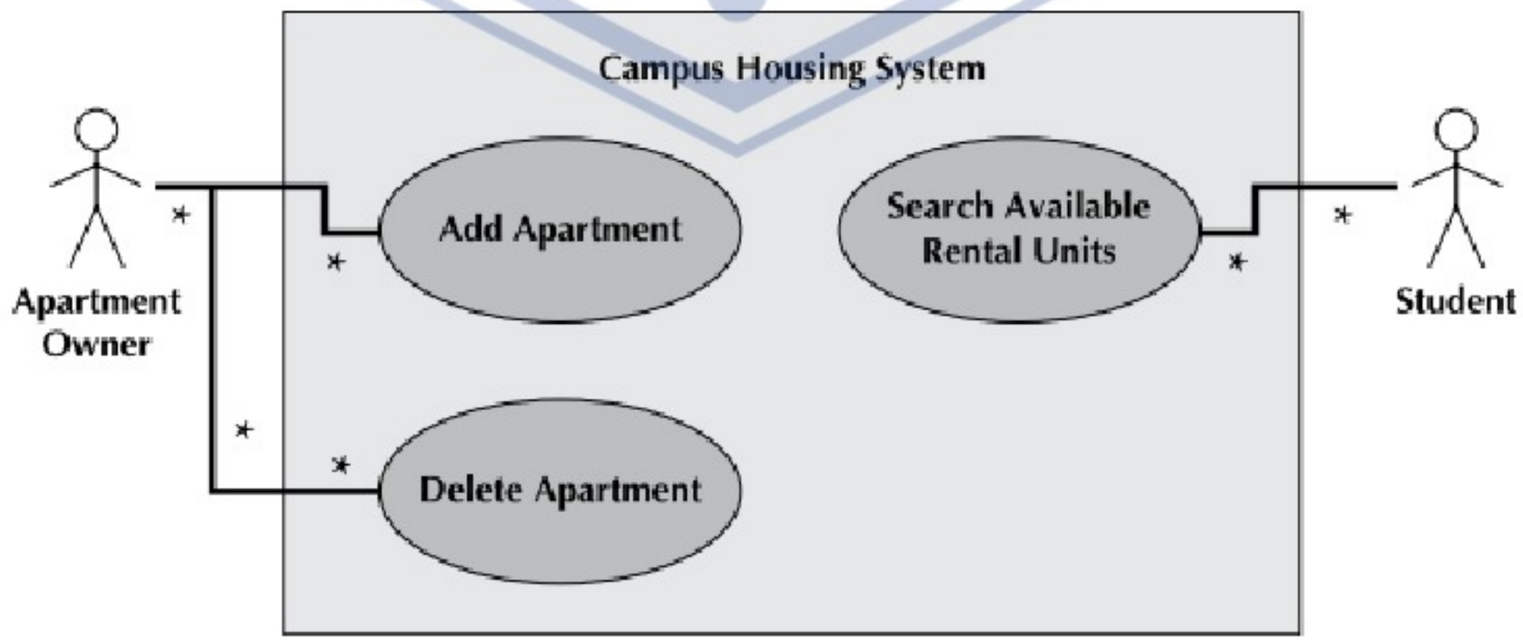
Dalam konteks pemodelan UML (*Unified Modeling Language*), *Use Case* menjadi fondasi bagi seluruh diagram lain. Semua teknik pemodelan dalam UML, seperti *sequence diagram*, *activity diagram*, dan *class diagram* dibangun berdasarkan *Use Case*. Dengan demikian, *Use Case* adalah titik awal dari seluruh rancangan sistem[10].

Use Case kemudian divisualisasikan dalam bentuk *Use Case Diagram*, yaitu diagram yang menggambarkan hubungan antara aktor dan fungsi-fungsi sistem (*Use Case*) yang dijalankan. Diagram ini menampilkan siapa saja yang menggunakan sistem serta aktivitas apa saja yang dapat dilakukan oleh masing-masing aktor. *Use Case Diagram* juga memudahkan komunikasi antara tim pengembang dan pengguna karena menggunakan simbol-simbol visual yang sederhana dan mudah dipahami. Berikut adalah simbol-simbol yang digunakan dalam *Use Case Diagram*[10]:

Tabel 2. 1 Simbol *Use Case Diagram*

No.	Nama Simbol	Simbol	Keterangan
1.	*Actor* (Aktor)	 Actor/Role <<actor>> Actor/Role	Aktor merupakan entitas luar, baik manusia maupun sistem, yang berinteraksi dengan sistem untuk mencapai tujuan tertentu.
2.	*Use Case*	 Use Case	*Use Case* merupakan fungsi utama yang disediakan oleh sistem untuk pengguna. Digambarkan berada di dalam batas sistem dan biasanya diberi nama dengan

			frasa kerja-kata benda yang menjelaskan aktivitasnya.
3.	*Subject Boundary* (Batas Subjek)		*Subject boundary* merupakan batas yang menunjukkan ruang lingkup sistem. Digambarkan sebagai sebuah kotak besar yang berisi nama sistem di bagian atas atau dalam kotak tersebut.
4.	*Association Relationship* (Asosiasi)		*Association relationship* merupakan garis penghubung antara aktor dan use case yang menunjukkan adanya interaksi atau komunikasi di antara keduanya.
5.	*Include Relationship* (Termasuk)		*Include relationship* menunjukkan suatu case menyertakan atau menggunakan fungsi dari *use case* lain sebagai bagian dari prosesnya.
6.	*Extend Relationship* (Perluasan)		*Extend relationship* menunjukkan perluasan fungsi dari suatu *use case* dengan menambahkan perilaku atau langkah opsional.
7.	*Generalization Relationship* (Generalisasi)		*Generalization relationship* menunjukkan hubungan pewarisan antara *use case* yang bersifat khusus dengan *use case* yang lebih umum.



Gambar 2. 3 Contoh *Use Case Diagram*

Tabel 2. 2 *Template* Deskripsi *Use Case*

***Use Case* Name**: Diisi dengan nama *Use Case*	**ID**: ID *Use Case*	**Important Level**: Tingkat penting dari *Use Case*
Primary Actor: Diisi dengan aktor utama	**Use Case Type**: Tipe dari *Use Case*	
Stakeholders and Interests: Diisi dengan siapa saja yang terlibat dalam *Use Case*		
Brief Description: Deskripsi dari tujuan dibuatnya deskripsi *Use Case*		
Trigger: Aksi yang dilakukan oleh aktor pada *Use Case*. **Type**: Tipe pemicu *Use Case*		
Relationships: Hubungan use case menjelaskan bagaimana use case terkait dengan *Use Case* lain dan pengguna.		
Association: Sebuah hubungan *association* menggambarkan komunikasi yang terjadi antara use case dan aktor yang menggunakan *Use Case* tersebut.		
Include: Hubungan include mewakili inklusi wajib dari *Use Case* lain.		
Extend: Hubungan *extend* mewakili perluasan fungsionalitas *Use Case* untuk mencakup aksi opsional.		
Generalization: Hubungan *Generalization* memungkinkan *Use Case* untuk mendukung pewarisan.		
Normal Flow of Events: Alur peristiwa normal hanya mencakup langkah-langkah yang biasanya dieksekusi dalam sebuah use case..		
Subsflows: Dalam beberapa kasus, alur peristiwa normal sebaiknya dibagi menjadi serangkaian subalur untuk menjaga agar alur peristiwa normal tetap sesederhana mungkin.		
Alternate/Exceptional Flows: Aliran alternatif adalah aliran yang memang terjadi tetapi tidak dianggap sebagai normal.		

Tabel 2. 3 Contoh Deskripsi *Use Case Add Apartment*

Nama *Use Case*: Tambahkan Apartemen	ID: 1	Important Level: Tinggi
Aktor Utama: Pemilik Apartemen	Tipe *Use Case*: Detail, Esensial	
Pihak berkepentingan: Pemilik Apartemen - Ingin mengiklankan kamar apartemen yang tersedia. Layanan Perumahan Kampus - Menyediakan layanan yang memungkinkan pemilik apartemen untuk menyewakan kamar apartemen yang tersedia.		
Ringkasan Singkat: *Use Case* ini menjelaskan bagaimana layanan perumahan kampus dapat mempertahankan daftar yang selalu diperbarui tentang kamar apartemen yang tersedia.		
Trigger: Pemilik apartemen ingin menambahkan kamar apartemen yang tersedia. Type: *External*		
Relationships:		
Association: Pemilik Apartemen		
Include: -		
Extend: -		
Generalization: -		

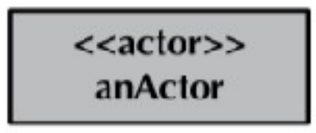
Normal Flow of Events: 1. Mencatat lokasi apartemen. 2. Mencatat jumlah kamar tidur di apartemen. 3. Mencatat sewa bulanan apartemen. 4. Menambahkan apartemen ke daftar apartemen yang tersedia.
Subsflows: -
Alternate/Exceptional Flows: -

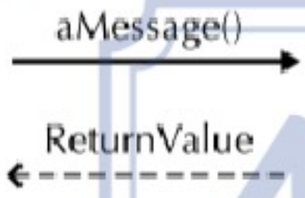
2.3.2 Sequence Diagram

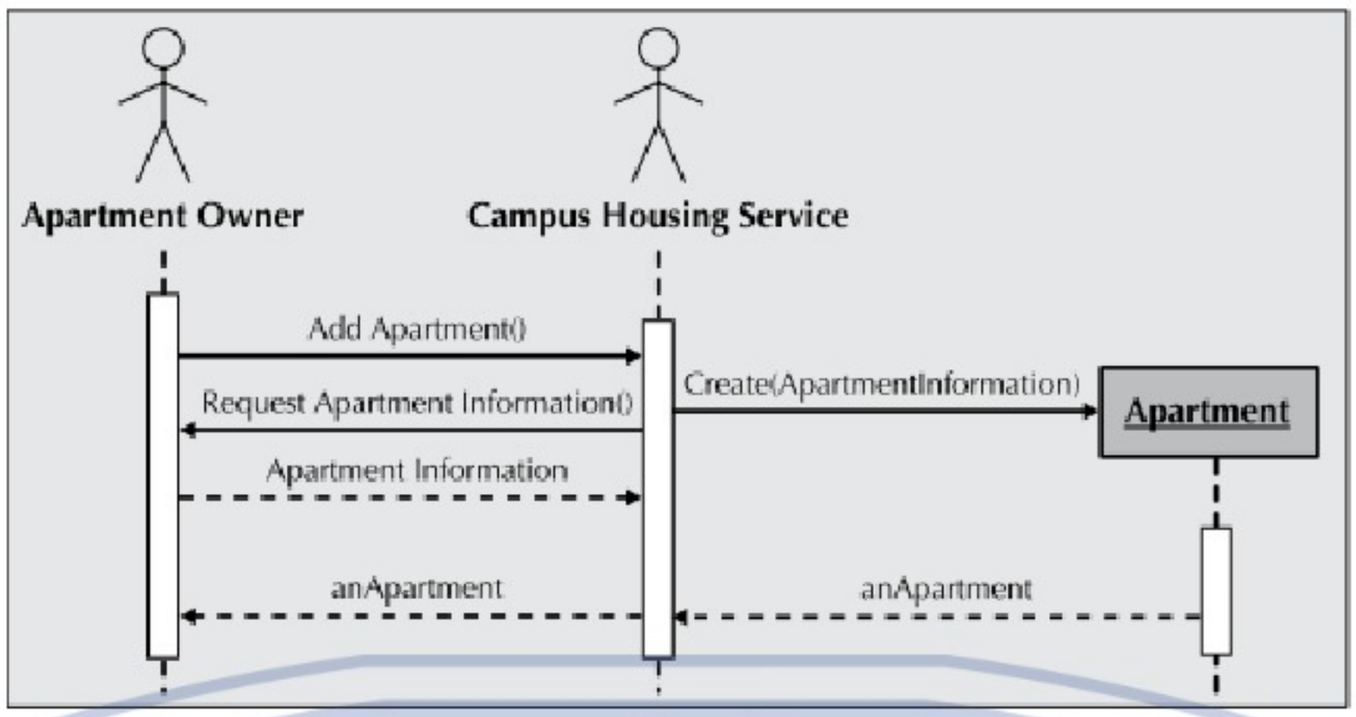
Sequence Diagram merupakan salah satu diagram dalam UML (*Unified Modeling Language*) yang digunakan untuk menggambarkan interaksi antara aktor dan objek dalam sebuah *Use Case*. Diagram ini menunjukkan bagaimana proses berlangsung langkah demi langkah dari awal hingga akhir suatu skenario. *Sequence Diagram* juga menampilkan aktor dan objek yang terlibat dalam proses dan bagaimana mereka saling berkomunikasi, sama seperti *Use Case Diagram*. Aktor dan objek diletakkan di bagian atas diagram, sedangkan waktu digambarkan secara vertikal dari atas ke bawah. Di bawah setiap aktor atau objek terdapat *lifeline*, yaitu garis putus-putus vertikal yang menunjukkan keberadaan entitas tersebut selama proses berlangsung[10].

Dengan demikian, *sequence diagram* berfungsi untuk memvisualisasikan urutan interaksi antara aktor dan objek secara kronologis, membantu analisis dan pengembang memahami alur logika sistem serta hubungan antar komponen yang terlibat di dalamnya. Berikut adalah simbol-simbol yang digunakan dalam *Sequence Diagram[10]*:

Tabel 2. 4 Simbol *Sequence Diagram*

No.	Nama Simbol	Simbol	Keterangan
1.	*Actor* (Aktor)	 anActor  <<actor>> anActor	Aktor merupakan entitas luar, baik manusia maupun sistem yang berinteraksi dengan sistem dan berperan dalam mengirim atau menerima pesan selama proses berlangsung.
2.	*Object* (Objek)	 anObject : aClass	*Object* merupakan elemen yang berpartisipasi dalam suatu *sequence* dengan cara mengirim dan/atau menerima pesan.

3.	*Lifeline* (Garis hidup)		*Lifeline* menunjukkan masa hidup atau keberadaan suatu objek selama urutan interaksi berlangsung. *Lifeline* berakhir dengan tanda X yang menandakan objek tersebut tidak lagi berinteraksi pada titik tersebut.
4.	*Execution Occurrence* (Eksekusi)		*Execution occurance* merupakan persegi panjang yang ditempatkan di atas *lifeline* dan digunakan untuk menunjukkan periode saat suatu objek sedang mengirim atau menerima pesan.
5.	*Message* (Pesan)		*Message* adalah komunikasi yang menyampaikan informasi dari satu objek ke objek lainnya. Sedangkan *return value* digambarkan dengan panah putus-putus menunjukkan hasil yang dikembalikan.
6.	*Guard Condition* (Penjagaan)		*Guard condition* merupakan syarat atau kondisi logis yang harus terpenuhi agar suatu pesan dapat dikirim.
7.	*Object Destruction* (X)		*Object destruction* digambarkan dengan tanda X di ujung *lifeline* suatu objek yang menandakan bahwa objek tersebut tidak lagi ada atau keluar dari sistem pada titik tersebut.
8.	*Frame* (Bingkai)		*Frame* digunakan untuk menunjukkan konteks atau batasan dari *sequence diagram*.



Gambar 2. 4 Contoh *Sequence Diagram*

2.3.3 Activity Diagram

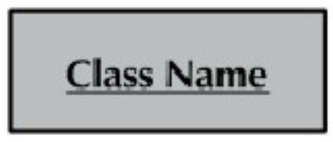
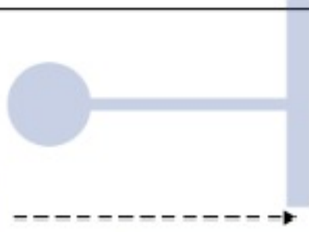
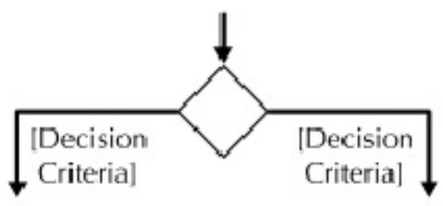
Activity Diagram merupakan salah satu diagram penting dalam pemodelan sistem yang berfungsi untuk menggambarkan alur logis dari suatu proses bisnis atau *workflow*. Diagram ini menunjukkan bagaimana aktivitas-aktivitas dalam sistem saling terhubung, dimulai dari awal proses hingga mencapai hasil akhir. Setiap aktivitas digambarkan menggunakan simbol berbentuk persegi panjang dengan sudut tumpul (*rounded rectangle*), yang mempresentasikan tindakan atau kegiatan tertentu dalam sistem[10].

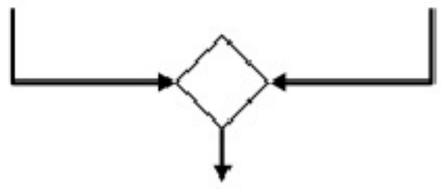
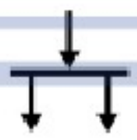
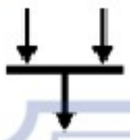
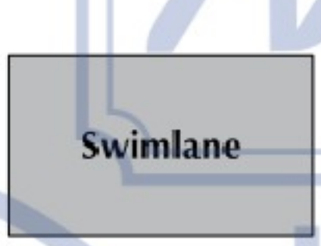
Dalam konteks pengembangan sistem berbasis *object-oriented*, *Activity Diagram* berperan untuk menghubungkan antara model bisnis dan model *Use Case*. Diagram ini membantu mendokumentasikan perilaku sistem secara logis tanpa merinci cara implementasinya. Diagram ini juga dapat mencakup elemen-elemen seperti keputusan (*decision node*), *fork* dan *join*, serta *swimlane* yang memperlihatkan tanggung jawab antar aktor dalam suatu proses. Dengan elemen-elemen ini, *activity diagram* menjadi alat visual yang efektif untuk menjelaskan bagaimana berbagai bagian dalam sistem bekerja sama mencapai tujuan tertentu[10].

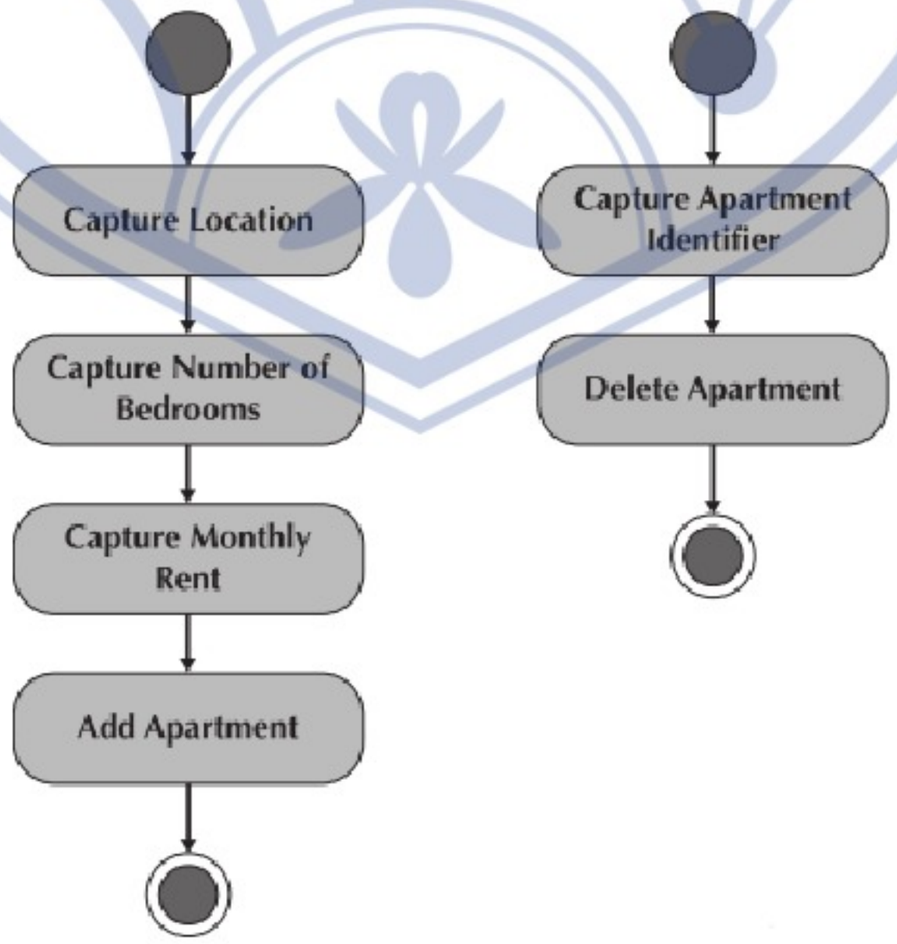
Simbol-simbol yang digunakan dalam *activity diagram* adalah sebagai berikut[10]:

Tabel 2. 5 Simbol *Activity Diagram*

No.	Nama Simbol	Simbol	Keterangan
1.	*Action* (Tindakan)	Action	*Action* merupakan perilaku sederhana yang tidak dapat diuraikan lagi menjadi aktivitas yang lebih kecil.

2.	*Activity* (Aktivitas)		*Activity* digunakan untuk mewakili sekumpulan a*ctions* yang saling berkaitan dalam suatu proses.
3.	*Object Node* (Simpul Objek)		*Object Node* digunakan untuk mewakili suatu objek yang terhubung dengan o*bject flow*.
4.	*Control Flow* (Aliran Kontrol)		*Control Flow* digunakan untuk menunjukkan urutan eksekusi. Memperlihatkan proses berpindah dari satu aktivitas ke aktivitas berikutnya.
5.	*Object Flow* (Aliran Objek)		*Object Flow* digunakan untuk menunjukkan aliran suatu objek dari satu aktivitas ke aktivitas lainnya. Menggambarkan bagaimana informasi berpindah sepanjang proses.
6.	*Initial Node* (Simpul Awal)		*Initial Node* digunakan untuk menunjukkan titik awal dari rangkaian aksi atau aktivitas.
7.	*Final-Activity Node* (Simpul Aktivitas Akhir)		*Final-Activity Node* digunakan untuk menghentikan seluruh *control flow* dan *object flow*, menandakan proses telah selesai sepenuhnya.
8.	*Final-Flow Node* (Simpul Aliran Akhir)		*Final-Flow Node* digunakan untuk menghentikan satu *control flow* atau *object flow* tertentu tanpa menghentikan keseluruhan proses atau aktivitas lainnya.
9.	*Decision Node* (Simpul Keputusan)		*Decision Node* digunakan untuk mewakili kondisi pengujian yang memastikan bahwa alur objek hanya mengikuti satu jalur yang sesuai dengan hasil pengujian tersebut.

10.	*Merge Node* (Gabungkan Node)		*Merge Node* digunakan untuk menggabungkan kembali beberapa jalur keputusan yang sebelumnya diupisahkan oleh *decision node*, sehingga alur proses dapat dilanjutkan ke langkah berikutnya.
11.	*Fork Node* (Simpul Garpu)		*Fork Node* digunakan membagi alur proses menjadi beberapa alur paralel, sehingga beberapa aktivitas dapat dijalankan secara bersamaan.
12.	*Join Node* (Gabung Node)		*Join Node* digunakan untuk menggabungkan kembali sejumlah alur paralel, sehingga proses dapat dilanjutkan setelah semua aktivitas paralel tersebut selesai dijalankan.
13.	*Swimlane*		*Swimlane* digunakan untuk membagi *Activty Diagram* menjadi baris atau kolom guna menunjukkan siapa (individu atau objek) yang bertanggung jawab menjalankan setiap aktivitas.



Gambar 2. 5 Contoh *Activity Diagram*

2.3.4 Class Diagram

Class Diagram merupakan model statis yang menggambarkan kelas-kelas beserta hubungan antar kelas yang tetap konsisten dalam suatu sistem dari waktu ke waktu. Diagram ini menunjukkan bagaimana setiap kelas memiliki atribut dan metode yang berperan dalam menjalankan proses dalam sistem. *Class Diagram* menampilkan hubungan antar kelas, seperti asosiasi dan generalisasi yang memperlihatkan bagaimana satu kelas berinteraksi terhadap kelas lainnya. Diagram ini berfokus pada struktur sistem secara keseluruhan, bukan urutan atau alur proses yang terjadi. *Class Diagram* menjadi pondasi penting dalam perancangan sistem berorientasi objek, karena memberikan gambaran lengkap mengenai elemen-elemen utama yang membentuk sistem[10].

Berikut adalah simbol-simbol yang digunakan dalam *Activity Diagram*[10]:

Tabel 2. 6 Simbol *Activity Diagram*

No.	Nama Simbol	Simbol	Keterangan
1.	*Class* (Kelas)	Class1 -Attribute-1 +Operation-1()	*Class* merupakan entitas utama yang mewakili orang, tempat, atau benda yang perlu disimpan informasinya ke dalam sistem.
2.	*Attribute* (Atribut)	attribute name /derived attribute name	*Attribute* menggambarkan properti atau karakteristik dari suatu objek.
3.	*Operation* (Operasi)	operation name ()	*Operation* menunjukkan tindakan atau fungsi yang dapat dilakukan oleh *class*, seperti membuat, memperbarui, mengambil, atau menghapus data. *Operation* diklasifikasikan sebagai *constructor*, *query*, *update*, atau *destructor*.
4.	*Association* (Asosiasi)	AssociatedWith 0..* 1	*Association* menggambarkan hubungan antara dua atau lebih kelas. Dilabeli dengan kata kerja atau nama peran yang paling mewakili keterkaitannya.

5.	*Generalization* (Generalisasi)		*Generalization* menunjukkan hubungan "*a-kind-of*" antara beberapa kelas.
6.	*Aggregation* (Agregasi)		*Anggregation* menggambarkan hubungan "*a-part-of*" secara logis antara beberapa kelas atau antara satu kelas dengan dirinya sendiri.
7.	*Composition* (Komposisi)		*Composition* menunjukkan hubungan "*a-part-of*" secara fisik antara beberapa kelas atau antara satu kelas dengan dirinya sendiri.

Gambar 2. 6 Contoh *Class Diagram*

2.4 Basis Data

Basis data merupakan sekumpulan data yang terstruktur dengan baik di dalam suatu sistem yang digunakan untuk keperluan akses informasi pada suatu organisasi. Basis data berfungsi untuk menyimpan, mengelola, dan mengakses data dalam satu tempat sehingga prosesnya menjadi efisien[11]. Dalam melakukan operasional basis data, diperlukan arsitektur basis data yang memberikan panduan tentang cara penggunaan dan pengelolaan basis data yang efektif dan efisien. Berikut komponen-komponen dari arsitektur basis data[11]:

1. Basis Data

   Basis data merupakan kumpulan data yang terstruktur dan terintegrasi di dalam sistem basis data.

2. Aplikasi Basis Data

   Aplikasi basis data merupakan program yang digunakan untuk melakukan operasi seperti pencarian data, pembaruan data, dan penghapusan data.

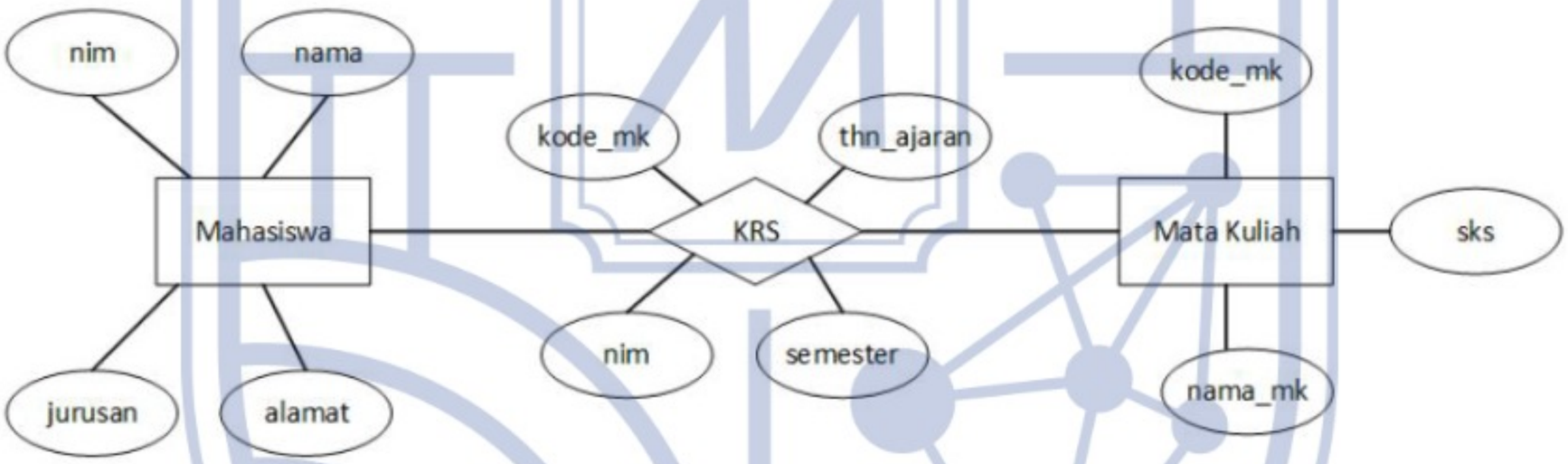
3. Pengguna Basis Data

Pengguna basis data merupakan individu atau kelompok yang berperan menggunakan basis data untuk tujuan tertentu, seperti pemrosesan data.

4. Perangkat Keras Dan Perangkat Lunak

   Perangkat keras dan perangkat lunak digunakan dalam arsitektur basis data mencakup *server*, *storage*, sistem manajemen basis data (DBMS), bahasa *query*, dan aplikasi yang berhubungan dengan basis data.

Dalam membantu pengembang perangkat lunak memahami, merancang, dan mengimplementasikan struktur database, basis data harus digambarkan dengan hubungan antar entitas, atribut, dan relasi. Entitas adalah objek yang diidentifikasi dan memiliki nilai yang akan disimpan ke dalam database. Atribut adalah properti dari suatu entitas yang memiliki karakteristik dari entitas tersebut, sedangkan relasi merupakan hubungan antara dua atau lebih entitas dalam sebuah database, seperti *one-to-one* (1:1), *one-to-many* (1:M), dan *many-to-many* (M:M). Konsep relasi antar entitas ini dikenal sebagai *Entity Relationship Diagram* (ERD)[11].



Gambar 2. 7 Contoh *Entity Relationship Diagram*

Gambar 2.7 di atas merupakan diagram relasi entitas dalam basis data, yang menunjukkan dua entitas (digambarkan dalam bentuk persegi panjang), yaitu: Mahasiswa dan Mata Kuliah. Selain itu, terdapat satu proses yang menghubungkan kedua entitas tersebut. Setiap entitas memiliki atribut yang khas, yaitu: nim, nama, jurusan, dan alamat untuk entitas Mahasiswa, nama_mk, kode_mk, dan sks untuk Mata Kuliah[11].

Untuk mengelola data dalam basis data, diperlukan suatu sistem yang disebut dengan Database Management System (DBMS). DBMS merupakan perangkat lunak yang digunakan untuk mengelola, menyimpan, memanipulasi, dan memelihara data[11].

Dalam arsitekturnya, DBMS memiliki tiga komponen utama, yaitu[12]:

1. Bahasa Definisi Data (*Data Definition Language*)

Data Definition Language (DDL) digunakan untuk menentukan struktur, tabel, kolom, tipe data, dan batasan-batasan data yang akan disimpan ke dalam *database*. DDL juga digunakan untuk membuat, mengubah, dan menghapus objek dalam *database*.

2. Bahasa Manipulasi Data (*Data Manipulation Language*)

   Data Manipulation Language (DML) digunakan untuk memanipulasi data dengan operasi CRUD (*Create*, *Read*, *Update*, *Delete*) di dalam *database*. Operasi ini memungkinkan pengguna untuk memasukkan, mengubah, melakukan pembaruan, dan menghapus data dalam *database*.

3. Bahasa Query (*Data Query Language*)

   Data Query Language memungkinkan pengguna untuk mengambil data dari basis data dengan melakukan kueri, serta melakukan pengurutan, pengelompokkan, dan penyeleksian data. DQL berperan untuk mengakses data dan mengekstrak informasi dari *database* yang diinginkan.

Dalam pengelolaan data, DBMS memiliki peran penting dalam menjaga integritas data dan mendukung proses pengambilan keputusan berbasis data yang akurat. Terdapat sepuluh fungsi utama DBMS, antara lain: menjaga integritas data, pengelolaan penyimpanan data, pengelolaan kamus data, transformasi serta penyajian data, pengamanan data, kemudahan akses data, penyediaan prosedur *backup* dan *recovery*, penyediaan bahasa akses dan pemrograman, antarmuka untuk komunikasi, serta manajemen transaksi[12].

2.5 *Enterprise Resource Planning* (ERP)

2.5.1 Konsep Enterprise Resource Planning (ERP)

Sistem *Enterprise Resources Planning* (ERP) dipecah menjadi tiga elemen utama. Yang pertama, *Enterprise* berasal dari bahasa inggris yang berarti perusahaan, mengacu pada kondisi bisnis yang mencakup berbagai sektor dan ukuran. *Enterprise* juga dapat diartikan sebagai individu maupun kelompok yang mengelola sumber daya dan berkolaborasi untuk mencapai tujuan bersama. Yang kedua, *Resources* berasal dari bahasa inggris yang berarti sumber daya, mengacu pada berbagai sumber daya yang dimiliki *Enterprise*, seperti aset, sumber daya manusia, teknologi, dan pelanggan yang dikelola dengan tujuan meningkatkan keuntungan. Elemen yang terakhir adalah *Planning*, berasal dari bahasa inggris yang berarti perencanaan, merupakan tahap perencanaan yang dilakukan perusahaan untuk memaksimalkan pemanfaatan *Resources* untuk mencapai target perusahaan[13].

Enterprise Resources Planning (ERP) terdiri dari kumpulan modul-modul untuk berbagai fungsi dan proses dalam perusahaan, seperti *Human Resources*, *Inventory*, *Sales & Marketing*, *Purchase*, dan *Finance & Accounting*. Dalam pengembangan sistem informasi ini, modul ERP yang dipakai berfokus pada modul *Sales*, *Purchase*, dan *Inventory*. Berikut penjabaran ketiga modul yang dipakai[14]:

1. *Sales*

   Proses penjualan secara umum mencakup proses seperti permintaan penjualan, menerima pesanan penjualan, menyusun faktur penjualan, dan pengiriman penjualan. Semua transaksi ini dikelola oleh modul penjualan ERP.

2. *Purchase*

   Modul pembelian (*Purchase*) mengurus semua proses pembelian yang mencakup pengadaan persediaan bahan baku. Modul pembelian berfungsi untuk menampung daftar pemasok, daftar menu, dan data pembelian. Modul pembelian terintegrasi dengan modul *Inventory* untuk memperbarui stok persediaan.

3. *Inventory*

   Modul persediaan (*Inventory*) digunakan untuk melacak stok persediaan bahan baku. Dengan menggunakan sistem persediaan yang berkode unik pada setiap item, pengguna dapat melacak item. Pelacakan berfungsi untuk mengetahui informasi stok persediaan bahan baku secara *real time*.

Proses bisnis perusahaan memerlukan sistem yang terintegrasi dan memberikan informasi yang relevan dan akurat. Melalui *Enterprise Resources Planning* (ERP), sebuah sistem terintegrasi dengan multimodul dirancang untuk melayani dan mendukung berbagai fungsi bisnis dan fungsi-fungsi yang memerlukan suatu kesatuan yang utuh sehingga memungkinkan data akan dibagi antara departemen yang berbeda. ERP digunakan untuk mengkoordinasikan informasi di setiap bisnis area, mengelola proses bisnis perusahaan, dan berbagi alat pelaporan manajemen[15].

2.5.2 Enterprise Resource Planning untuk UMKM

Usaha Mikro, Kecil, dan Menengah (UMKM) merupakan kelompok usaha yang dijalankan secara individu, rumah tangga, maupun badan usaha berskala kecil. Walaupun berskala kecil, UMKM berperan dalam mendorong perekonomian negara. Seiring pertumbuhan UMKM di Indonesia yang pesat, memaksa para pelaku UMKM untuk dapat beradaptasi dan mengambil strategi untuk meningkatkan daya saing antar sesama pelaku

UMKM. Salah satu strategi yang diambil adalah mengimplementasikan sistem terintegrasi yang berperan dalam keberlangsungan UMKM[16].

UMKM yang masih melakukan proses secara manual tanpa menggunakan sistem akan sangat sulit bagi pelaku UMKM. Dengan mengimplementasikan ERP, UMKM dapat meningkatkan produktivitas melalui proses bisnis yang terintegrasi. ERP membantu mengoptimalkan proses bisnis dengan mengimplementasikan modul ERP seperti *sales*, *purchase*, dan *inventory* sehingga dapat berjalan lebih efisien dan efektif. Di sisi lain, ERP dapat meningkatan visibilitas ke dalam proses bisnis, sehingga dapat menyediakan akses informasi secara *real time* dan meningkatkan daya saing. Menerapkan *Enterprise Resources Planning* (ERP) di UMKM dapat memberikan dampak positif yang signifikan terhadap produktivitas. Di era persaingan yang semakin ketat, sebuah sistem yang terintegrasi akan membantu usaha besar dan kecil menyederhakan proses transaksi yang terjadi di UMKM[17].

2.6 Bisnis Catering

Bisnis berasal dari bahasa inggris *business*, yang merupakan kegiatan yang dilakukan pereorangan maupun badan usaha berskala keci yang terlibat dalam kegiatan penjualan, pembelian, dan pembelian. Bisnis bertujuan untuk menyediakan hasil akhir yang didapat dari pencapaian tujuan oleh para pelaku bisnis. Bisnis tidak hanya menyediakan menu, tetapi juga layanan yang diinginkan oleh pelanggan.

Salah satu layanan bisnis adalah bisnis katering. Katering adalah salah satu jenis bidang makanan yang berbentuk makanan sudah jadi dan diantar langsung ke tempat pemesanan pada alamat pengiriman. Industri katering didorong untuk menggunakan konsep baru sebagai cara mempertahankan biaya tetap mereka seperti harga bahan baku, selain mengoptimalkan biaya lainnya. Hal ini mengharuskan bisnis katering harus dapat meningkatkan daya saing ditengah peningkatan pesaing pelaku bisnis katering yang lain. Oleh karena itu, pengimplementasian ERP terhadap bisnis katering akan sangat membantu dalam menciptakan inovasi dan strategi model bisnis katering[18].