BAB II
TINJAUAN PUSTAKA

2.1. Studio Musik

Studio musik merupakan tempat dimana para musisi, guru/dosen, para psikolog/terapis, sampai para sutradara melakukan rekaman musik. Selain untuk tempat perekaman, studio musik juga memiliki fungsi turunan. Studio musik juga bisa digunakan bagi mereka yang memiliki minat atau hobi di bidang musik, tapi memiliki keterbatasan alat yang dimiliki. Mereka bisa berlatih dengan menggunakan alat-alat musik yang disediakan oleh pihak studio musik dengan cara menyewa.[4]

Dalam mendirikan usaha studio musik, perlu diperhatikan pemilihan lokasi yang strategis untuk mendirikan usaha tersebut. Lokasi yang berada dekat dengan sekolah, universitas, atau instansi perkantoran menjadi prospek yang baik dalam pemilihan lokasi agar calon pelanggan mudah mengetahui keberadaan studio musik tersebut. Selain pemilihan lokasi, manajemen pemasaran suatu usaha studio musik juga perlu dilakukan agar studio musik tersebut dikenal oleh masyarakat.[4]

Pemasaran ini bisa dilakukan dengan cara konvensional, seperti memasang iklan, penyebaran brosur/pamflet, atau dengan cara-cara yang lebih 'kekinian', seperti lewat media-media sosial (*Facebook, Twitter, YouTube,* dll.). Sedangkan materi-materi promosi yang disampaikan adalah layanan dari studio musik tersebut, sampai pada fasilitas alat-alat apa saja yang bisa digunakan oleh konsumen.[4]

2.2. Analisis dan Perancangan Sistem Informasi

Analisis dan perancangan sistem adalah proses mengembangkan sebuah logika sistem informasi baru secara efektif menggunakan perangkat keras, perangkat lunak, data, proses, dan orang-orang untuk mendukung tujuan bisnis perusahaan. Tujuan dari analisis sistem adalah untuk memahami usulan aplikasi yang akan dirancang, memastikan bahwa usulan sistem tersebut telah memenuhi kebutuhan calon pengguna, dan memberikan rancangan sistem yang baik untuk dikembangkan.[5]

Analisis sistem mencakup empat kegiatan utama, antara lain[5]:

1. *Requirements modeling*

   Pemodelan persyaratan merupakan kegiatan pencarian fakta untuk mendeskripsikan sistem yang sudah ada dan mengidentifikasi persyaratan yang dibutuhkan untuk merancang sistem baru. Fakta-fakta tersebut dapat berupa[5]:

   a. *Output,* mengacu pada informasi elektronik atau cetak yang dihasilkan oleh sistem.
   b. *Input,* mengacu pada data yang diperlukan yang masuk ke sistem, baik secara manual atau otomatis.
   c. *Process*, mengacu pada aturan logis yang diterapkan untuk mengubah data menjadi informasi yang bermakna.
   d. *Performance,* mengacu pada karakteristik sistem, seperti kecepatan, volume, kapasitas, ketersediaan, dan keandalan.
   e. *Security*, mengacu pada perangkat keras, perangkat lunak, dan kontrol prosedural yang menjaga dan melindungi sistem dan datanya dari ancaman internal atau eksternal.

2. *Data and process modeling*

   Pemodelan data dan proses merupakan tahap lanjutan proses pemodelan dengan menunjukkan bagaimana merepresentasikan data dan proses sistem secara grafis menggunakan teknik analisis terstruktur tradisional. Analisis terstruktur mengidentifikasi data yang mengalir ke dalam proses, aturan bisnis yang mengubah data, dan aliran data keluaran yang dihasilkan.

3. *Object modeling*

   Pemodelan objek, merupakan kegiatan analisis berorientasi objek. Kegiatan ini menggabungkan data dan proses yang bekerja pada data menjadi hal-hal yang disebut objek. Objek-objek ini mewakili orang, benda, transaksi, dan peristiwa aktual yang mempengaruhi sistem. Selama proses pengembangan sistem, analis sering menggunakan kedua metode pemodelan untuk mendapatkan informasi sebanyak mungkin.

4. *Development strategies*

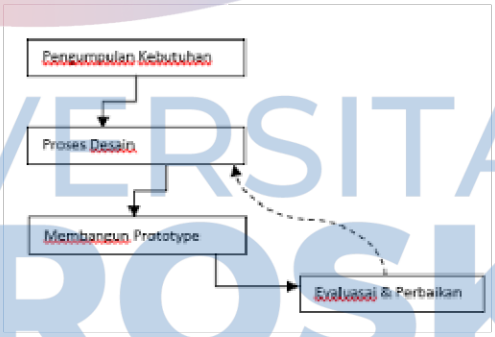
   Tahapan ini mempertimbangkan berbagai opsi pengembangan dan mempersiapkan transisi ke fase desain sistem.

Hasil, atau produk akhir, dari fase analisis sistem adalah dokumen persyaratan sistem, yang merupakan desain keseluruhan untuk sistem baru. Selain itu, setiap aktivitas dalam fase analisis sistem memiliki produk akhir dan satu atau beberapa pencapaian. [5]

2.2.1. Metode Prototyping

Prototyping adalah metode pengembangan sistem informasi yang membuat antara pengembang dan calon pengguna saling berinteraksi selama proses pengembangan sistem.[6] Jika pengguna dapat berinteraksi langsung dengan perancang aplikasi, maka setiap ide yang ada akan lebih mudah dipahami oleh pengguna.

Metode *prototyping* yang dilakukan bertujuan untuk menciptakan gambaran awal aplikasi yang ingin dikembangkan, rancangan *prototype* tersebut akan dievaluasi oleh user. Aplikasi *prototype* yang telah dievaluasi oleh pengguna akan menjadi rujukan dalam membangun aplikasi tersebut, yang akan menjadi produk akhir sebagai output dari penelitian ini. *Prototype* adalah modal yang sangat baik untuk dibawa ke presentasi apa pun.[7]



Gambar 2. 1. Ilustrasi Fase-Fase Prototype

Pada gambar di atas, dapat dilihat bahwa dalam metode pengembangan *prototyping* ini terdiri dari 4 fase, antara lain[3] [8]:

1. Pengumpulan kebutuhan

   Dalam mengumpulkan data yang akan dibutuhkan, ada beberapa cara yang sering digunakan antara lain[8][9]:

   a. Observasi

      Observasi merupakan salah satu teknik pengumpulan data dengan cara mengamati langsung objek data tersebut.

b. Wawancara

Wawancara adalah salah satu teknik mengumpulkan data dengan cara mempertemukan kedua belah pihak baik peneliti maupun subjek kajian, dan berinteraksi langsung untuk mendapatkan data yang dibutuhkan.

c. Studi Kasus

Teknik pengumpulan data yang dilakukan dengan cara mengumpulkan data-data dari sumber seperti buku, skripsi, jurnal, sehingga dapat dijadikan sebagai bahan acuan dalam penelitian.

Dalam tahap ini penulis melakukan pengumpulan data dengan cara wawancara secara langsung dengan pengguna sistem, dan observasi dengan mengamati aplikasi sejenis lainnya untuk mengetahui kebutuhan pengguna aplikasi ini.

Ada beberapa cara dalam melakukan pengumpulan data untuk kebutuhan sistem, dalam hal ini penulis akan melakukan pengumpulan data dengan melakukan wawancara. Wawancara merupakan salah satu teknik yang paling cepat untuk mendapatkan data-data yang dibutuhkan. Penulis akan melakukan wawancara terhadap dua pihak yaitu pemilik studio, dan pemain musik.

Dalam tahap ini, dengan menggunakan *software* Visio 2013 dan StarUML maka akan menghasilkan keluaran berupa:

a. *Use case diagram* yang menggambarkan perancangan aplikasi Simfoni.
b. *Class diagram* yang menggambarkan hubungan antara kelas-kelas yang terdapat serta menjadi dasar perancangan *database* dalam aplikasi Simfoni. Perancangan aplikasi ini akan menggunakan *mySQL* sebagai *database*.
c. *Sequence diagram* yang menggambarkan urutan interaksi setiap entitas.
d. *Activity diagram* yang menggambarkan aliran proses bisnis langkah demi langkah.

2. Proses desain yang cepat

Proses desain yang cepat merupakan tahapan merancang desain aplikasi yang mencakup input, proses, dan format output yang akan direpresentasikan kepada pengguna. Desain berfokus pada konsep aplikasi dari sisi pengguna yang meliputi *input*, proses, dan *output*.

Tahap ini akan menghasilkan keluaran berupa sketsa dari aplikasi secara global atau gambaran kasar dari tampilan aplikasi di bagian pemilik studio, pemain musik, dan admin yang dirancang menggunakan *software Balsamiq Mockups 3*.

3. Membangun *Prototype*

Setelah berhasil menciptakan desain aplikasi, langkah selanjutnya adalah membangun sebuah *prototype* yang diciptakan berdasarkan desain yang sudah dirancang. Tahapan ini akan menghasilkan keluaran berupa *prototype* dari gambaran interfaces aplikasi. Pada tahapan ini penulis akan membangun *prototype* menggunakan aplikasi Adobe XD.

4. Evaluasi dan Perbaikan

Prototype yang sudah dibangun akan dievaluasi kembali oleh pengguna dan disesuaikan dengan kebutuhan perangkat lunak yang akan dikembangkan. *Prototype* dikaji untuk memenuhi kebutuhan pengguna, dan membantu pengembang memahami secara lebih jelas apa yang akan dilakukan. Dalam mengevaluasi *prototype* yang telah dirancang penulis akan menggunakan metode *usability testing* dengan menggunakan kuesioner.

Usability testing merupakan pengujian antarmuka pengguna yang dikombinasikan dengan pengujian *use case,* dan berfokus pada seberapa baik fungsionalitas antarmuka pengguna mendukung *use case*. Seberapa efektif dan efisien rancangan antarmuka pengguna.

Tahapan ini memiliki keluaran berupa hasil akhir dari *prototype* yang telah dirancang, dievaluasi, dan diuji pengguna.

2.3. Aplikasi *Mobile*

Aplikasi *mobile* adalah sebuah perangkat lunak yang dapat digunakan oleh setiap pengguna dengan bebas, mudah, dan dalam keadaan berpindah-pindah/bergerak dengan menggunakan media telepon genggam atau seluler. Aplikasi *mobile* merupakan revolusi dari piranti lunak yang biasanya berbasis PC Desktop. Aplikasi *mobile* biasanya digunakan pada platform android dan iOS.[10]

2.3.1. Karakteristik Aplikasi *Mobile*

Aplikasi *mobile* memiliki karakteristik sebagai berikut[11][12]:

1. *Ubiquity*

   Ubiquity merupakan situasi/keadaan yang memungkinkan pengguna perangkat *mobile* untuk dapat menggunakan perangkatnya setiap saat kapanpun dan dimanapun.

2. *Privacy and Security*

   Pengguna membutuhkan kenyamanan dan kemudahan dalam melakukan transaksi, selain itu keamanan informasi data pribadi pengguna *mobile device* juga harus terjaga. Keamanan yang dijaga ketat oleh pihak internet banking maupun pihak *e-marketplace* sekalipun, tidak menutup kemungkinan diretas dengan mudah oleh pihak-pihak yang tidak bertanggung jawab demi menguntungkan diri sendiri. Oleh karena itu, *user* disarankan selalu berhati-hati saat menggunakan informasi pribadi.

3. *Mobility*

   Mobility artinya aplikasi dapat digunakan dalam sistem operasi apa saja, tanpa adanya ketentuan spesifik tentang jenis sistem operasi.

4. Mudah Digunakan

   Aplikasi *mobile* memiliki karakteristik *user-friendly*, maksudnya aplikasi tersebut dapat dengan mudah dipahami oleh pengguna tanpa harus mempelajari aplikasi tersebut. Pengguna dapat memahami penggunaan aplikasi tersebut hanya dengan melihat fitur-fiturnya saja.

5. *Reability*

   Aplikasi *mobile* bekerja sesuai dengan fungsi dan tujuan aplikasi tersebut, dan harus dapat diandalkan.

2.3.2. Jenis-jenis Aplikasi *Mobile*

Ada beberapa jenis aplikasi *mobile*, antara lain[13]:

1. *Native Mobile Application*

   Native Application adalah sebuah aplikasi *mobile* yang dikembangkan dengan ketentuan spesifik platform atau perangkat *mobile* yang akan digunakan. Aplikasi ini juga dikembangkan menggunakan bahasa pemrograman tertentu.

2. *Web Application*

   Aplikasi web merupakan aplikasi yang dapat dijalankan di perangkat *mobile* apapun menggunakan *browser*. Aplikasi ini dikodekan menggunakan bahasa HTML5, JavaScript, atau CSS. Aplikasi web ini juga tidak memiliki ketentuan spesifik sistem operasi, perangkat *mobile*, atau platform khusus untuk menjalankannya.

3. *Hybrid Mobile Application*

   Aplikasi ini merupakan gabungan dari fitur-fitur aplikasi web dan aplikasi *native.* Aplikasi ini memiliki performa kinerja layaknya aplikasi web, namun memiliki karakteristik seperti aplikasi *native.*

2.3.3. Keuntungan Aplikasi *Mobile*

Perkembangan teknologi yang begitu pesat saat ini, memberikan banyak nilai positif bagi aplikasi *mobile*. Berikut ini adalah beberapa keuntungan aplikasi *mobile*[14]:

1. Fondasi aplikasi seluler dibangun di atas protokol yang sudah ada dan populer. Secara khusus, penggunaan HTTP diadopsi secara luas dalam penerapan *mobile* dan dipahami dengan baik oleh pengembang.
2. Kemajuan teknis *smartphone* telah memungkinkan aplikasi *mobile* menawarkan fitur yang lebih canggih dan pengalaman pengguna yang lebih baik. Perbaikan dalam resolusi layar dan tampilan layar sentuh telah menjadi faktor utama dalam meningkatkan pengalaman pengguna interaktif, terutama dalam aplikasi permainan.
3. Peningkatan dalam masa pakai baterai dan daya pemrosesan memungkinkan *smartphone* modern untuk menjalankan tidak hanya satu tetapi banyak aplikasi sekaligus dan lebih lama. Ini sangat memudahkan pengguna akhir karena mereka memiliki satu perangkat yang dapat melakukan banyak fungsi.
4. Perbaikan dalam teknologi jaringan seluler telah menghasilkan peningkatan kecepatan yang signifikan. Secara khusus, cakupan 3G dan 4G yang luas telah memungkinkan pengguna untuk memiliki akses internet berkecepatan tinggi dari

smartphone mereka. Aplikasi *mobile* telah memanfaatkan hal ini sepenuhnya untuk menyediakan akses ke berbagai layanan *online*.

5. Kesederhanaan teknologi inti dan bahasa yang digunakan dalam pengembangan seluler telah membantu revolusi *mobile*. Aplikasi dapat dikembangkan dengan menggunakan bahasa populer seperti Java, yang dipahami dengan baik dan memiliki basis pengguna yang besar.

2.4. Unified Modelling Language (UML)

UML merupakan bahasa pemodelan yang merepresentasikan tujuan umum dari sebuah sistem secara menyeluruh tanpa batasan sistem tertentu. UML menyajikan konsep bahasa pemodelan dengan menggabungkan banyak elemen grafis untuk pemodelan berbagai area aplikasi, sehingga memudahkan dalam menentukan, merancang, memvisualisasikan, dan mendokumentasikan perkembangan rancangan suatu sistem.[15]

UML tidak terikat pada alat pengembangan tertentu, bahasa pemrograman tertentu, atau platform target khusus di mana sistem yang akan dikembangkan harus digunakan. UML dapat digunakan dalam setiap tahap pengembangan suatu sistem. Pada semua tahap perkembangan, konsep bahasa yang sama dapat digunakan dalam notasi yang sama. Dengan demikian, suatu model dapat disempurnakan secara bertahap. Diagram UML dibedakan menjadi dua, *structure diagrams* dan *behavior diagrams*. [15]

a. *Structure Diagrams*

1. *Class diagram,* diagram pemodelan data konseptual dan pengembangan perangkat lunak berorientasi objek.
2. *Object diagram,* merepresentasikan gambaran nyata dari status suatu sistem pada waktu eksekusi tertentu.
3. *Component diagram,* menggambarkan hubungan komponen yang menyediakan layanan bagi komponen lain, atau menggunakan layanan komponen lain.
4. *Deployment diagram,* menggambarkan topologi perangkat keras yang digunakan dan sistem yang berjalan.

b. *Behavior Diagrams*

1. *Activity diagram,* menggambarkan alur kerja serta aliran data yang mengkoordinasikan tindakan dan membentuk sebuah aktivitas.
2. *Use case diagram,* menjelaskan pengguna mana yang menggunakan fungsionalitas yang mana.
3. *Sequence diagram,* menggambarkan hubungan antara objek untuk menyelesaikan suatu tugas.
4. *Communication diagram,* mirip dengan *sequence diagram* namun diagram ini lebih berfokus menjelaskan siapa yang berinteraksi dengan siapa.
5. *The state machine diagram,* menggambarkan perilaku yang diizinkan dari suatu objek dalam bentuk kemungkinan status dan transisi status yang dipicu oleh berbagai peristiwa.

2.4.1. Use Case Diagram

Use case diagram adalah diagram yang mengilustrasikan dengan cara yang sangat sederhana fungsi utama sistem dan berbagai jenis pengguna yang akan berinteraksi.[16] *Use case diagram* juga memodelkan pengguna sistem mana yang menggunakan fungsionalitas mana, menyatakan siapa yang akan benar-benar bekerja dengan sistem yang akan dibangun. *Use case diagram* juga dapat digunakan untuk mendokumentasikan fungsionalitas sistem yang ada dan mencatat secara retrospektif pengguna mana yang diizinkan untuk menggunakan fungsionalitas mana.[15]

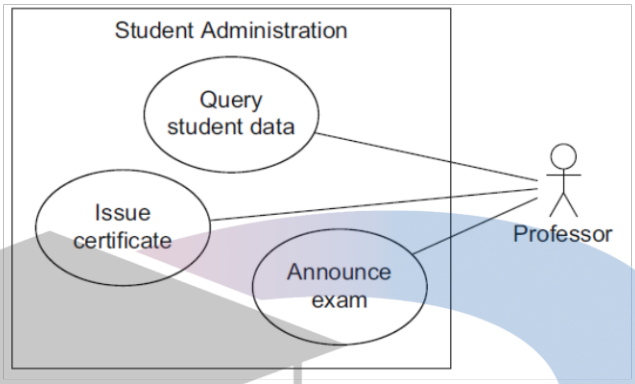
Use case biasanya direpresentasikan dalam bentuk elips. Nama *use case* dituliskan langsung di dalam atau di bawah elips. Alternatifnya, *use case* dapat direpresentasikan dengan persegi panjang yang berisi nama *use case* di tengah dan elips kecil di pojok kanan atas. Persegi panjang ini melambangkan batas-batas sistem yang akan dideskripsikan.[15]

Contoh pada Gambar 2.2 menunjukkan sistem Administrasi Mahasiswa, yang menawarkan tiga kasus penggunaan:

1. Menanyakan data siswa
2. Menerbitkan sertifikat

3. Mengumumkan ujian.

Kasus penggunaan ini mungkin dipicu oleh aktor Profesor. Berikut ini merupakan contoh representasi dari *use case*[15]*:*



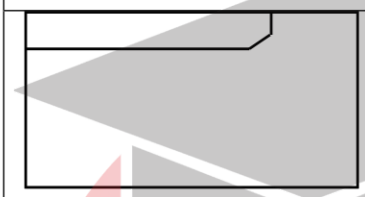
Gambar 2. 2. Representasi *Use Case*

Use Case Diagram terdiri dari beberapa elemen, antara lain:[16]

Tabel 2. 1. Elemen *Use Case Diagram*

Gambar	Nama	Keterangan
	Aktor	Menggambarkan orang yang sedang berinteraksi dengan aplikasi.
	*Use Case*	Menggambarkan fungsionalitas utama dari aplikasi, aksi-aksi apa saja yang dapat dilakukan oleh aktor
	Generalization	Merepresentasikan kasus penggunaan khusus ke yang lebih umum. Merupakan tanda panah yang ditarik dari kasus penggunaan khusus ke kasus penggunaan dasar.
	Include	Menspesifikasikan secara akurat bahwa *use case* menjadi sumber.

Tabel 2. 2. Elemen *Use Case Diagram(sambungan)*

<<extend>>	*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada satu titik yang diberikan.
	Association	Apa yang menghubungkan antara objek satu dengan objek lainnya.
	*Package*	Kumpulan dari *use case* / proses

Apabila suatu *use case* mencakup sangat banyak aktivitas yang terjadi, sehingga menjadikan *use case* tersebut tidak dapat digambarkan dalam satu halaman. Hal ini dapat diatasi dengan menggambarkan *use case* menggunakan narasi *use case.*[15] Narasi *use case* menceritakan bagaimana suatu sistem dan para penggunanya saling berinteraksi.

Alistair Cockburn menciptakan sebuah template terstruktur untuk narasi *use case* yang berisi informasi berikut[15]:

1. *Name*
2. *Short description*
3. *Precondition*: prasyarat agar eksekusi berhasil
4. *Postcondition*: status sistem setelah eksekusi berhasil
5. *Error situations*: kesalahan relevan dengan domain masalah
6. Status sistem saat terjadi kesalahan
7. Aktor yang berkomunikasi dengan *use case*
8. *Trigger*: peristiwa yang memulai / memulai *use case*
9. *Standard process*: langkah-langkah individu yang harus diambil
10. *Alternative processes*: penyimpangan dari proses standar

Contoh narasi *use case*[15]:

Tabel 2. 3. *Narrative Use Case for Reserve Lecture Hall*

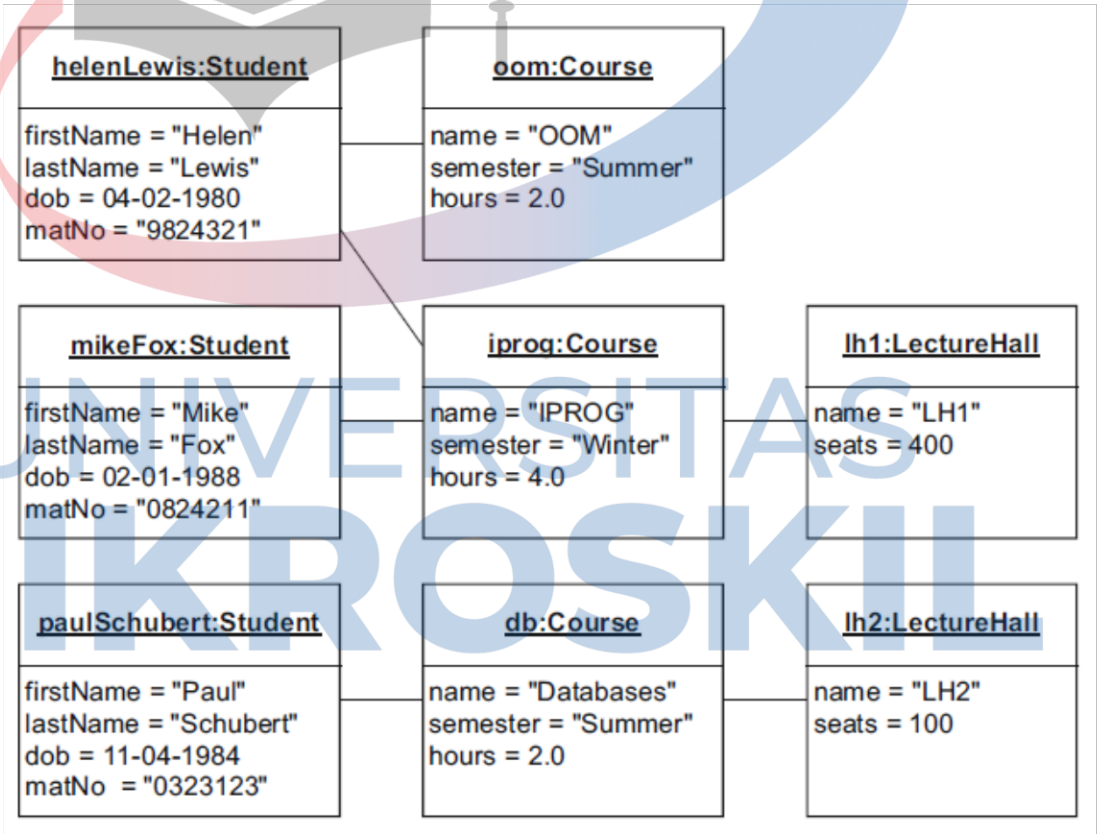
Name:	*Reserve lecture hall*
Short description	*An employee reserves a lecture hall at the university for an event.*
Precondition	*The employee is authorized to reserve lecture halls.* *Employee is logged in to the system.*
Postcondition	*A lecture hall is reserved.*
Error situations	*There is no free lecture hall.*
System state in the event of an error:	*The employee has not reserved a lecture hall.*
Actors:	*Employee*
Trigger:	*Employee requires a lecture hall.*
Standard Process:	*(1) Employee selects the lecture hall.* *(2) Employee selects the date.* *(3) System confirms that the lecture hall is free.* *(4) Employee confirms the reservation.*
Alternative Process:	*(3') Lecture hall is not free.* *(4') System proposes an alternative lecture hall.* *(5') Employee selects the alternative lecture hall and confirms the reservation.*

2.4.2. Class Diagram

Class diagram menggambarkan struktur sistem perangkat lunak dan dengan demikian membentuk notasi inti yang pertama kali dibahas untuk pemodelan berorientasi objek.[17] *Class diagram* biasanya untuk memodelkan struktur statis suatu sistem, sehingga diagram kelas menggambarkan elemen-elemen sistem dan hubungan di antara mereka. Elemen-elemen ini dan hubungan di antara mereka tidak berubah seiring waktu. Misalnya, mahasiswa memiliki nama dan nomor matrikulasi serta mengikuti berbagai mata kuliah. Kalimat ini mencakup sebagian kecil dari struktur universitas dan tidak kehilangan validitasnya bahkan selama bertahun-tahun. Hanya siswa dan mata kuliah tertentu yang berubah.[15]

2.4.2.1. Object

Object adalah representasi dari entitas yang nyata ataupun konsep dengan batasan-batasan yang tepat. Dalam diagram objek, sebuah objek ditampilkan sebagai persegi panjang yang dapat dibagi lagi menjadi beberapa kompartemen. Kompartemen pertama selalu berisi informasi dalam bentuk objectName: Class. Suatu sistem berisi banyak objek yang berbeda. Setiap objek dapat diidentifikasi secara unik. Misalnya, pada gambar menjelaskan sebagai bagian dari program Studi TI, Helen Lewis menghadiri kuliah Pemodelan Berorientasi Objek (OOM) di universitas. Helen Lewis, Kajian IT, dan Pemodelan Berorientasi Objek merupakan individu (objek konkrit) dalam suatu sistem administrasi universitas dan berada dalam hubungan satu sama lain. [15]

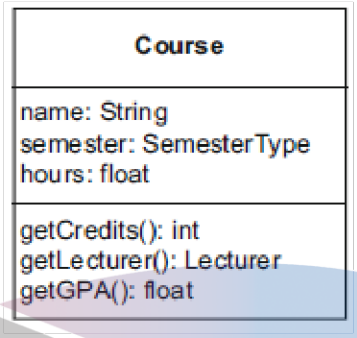


Gambar 2. 3. Contoh *Class Diagram.*

2.4.2.2. Class

Class adalah rencana konstruksi untuk sekumpulan objek serupa yang muncul di sistem yang akan ditentukan. *Class* dapat mencirikan, misalnya, orang (misalnya,

siswa), benda (misalnya, gedung), acara (misalnya, kursus atau ujian), atau bahkan konsep abstrak seperti kelompok. Gambar 2.5 menampilkan contoh *class*.[15]



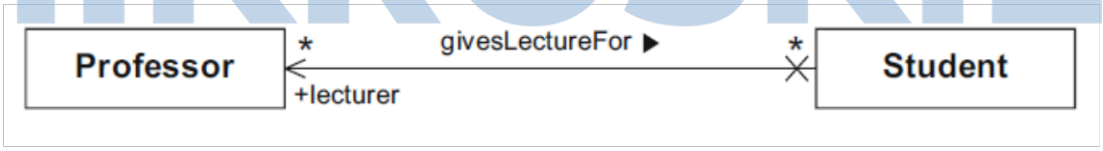
Gambar 2. 4. Contoh *Class*

2.4.2.3. Associations

Associations antar kelas memodelkan hubungan yang mungkin, yang dikenal sebagai tautan, antara instansi kelas. *Associations* menggambarkan kelas mana yang merupakan mitra komunikasi potensial. Jika atribut dan operasinya memiliki visibilitas yang sesuai, mitra komunikasi dapat mengakses atribut dan operasi satu sama lain.[15]

1. *Binary Associations*

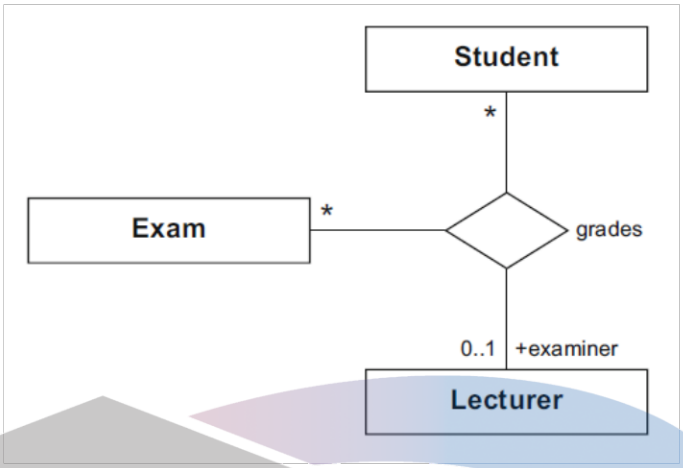
Binary associations menghubungkan dua kelas satu sama lain. Hubungan tersebut ditampilkan sebagai garis penuh antara kelas mitra yang terlibat. Garis tersebut dapat diberi label dengan nama asosiasi secara opsional diikuti dengan arah membaca, segitiga hitam kecil. Arah membaca diarahkan ke salah satu ujung. [15]



Gambar 2. 5. Contoh Asosiasi *Binary*

2. *N-Ary Associations*

N-Ary Associations menghubungan lebih dari dua *class*. Asosiasi n-ary diwakili dengan berlian berlubang di tengahnya. Berlian terhubung dengan semua mitra hubungan melalui garis yang tidak diarahkan. Nama asosiasi ditentukan di sebelah berlian. Tidak ada petunjuk navigasi untuk asosiasi n-ary. [15]



Gambar 2. 6. Contoh *N-Ary Associations*

3. *Aggregations*

Agregasi adalah bentuk asosiasi khusus yang digunakan untuk menyatakan bahwa instansi dari satu kelas adalah bagian dari instansi kelas lain. Pada prinsipnya, agregasi bersama mengungkapkan lemahnya kepemilikan bagian-bagian menjadi keseluruhan, yang berarti bahwa bagian-bagian juga ada secara independen dari keseluruhan.[15]

Gambar 2. 7. Contoh *Aggregations*

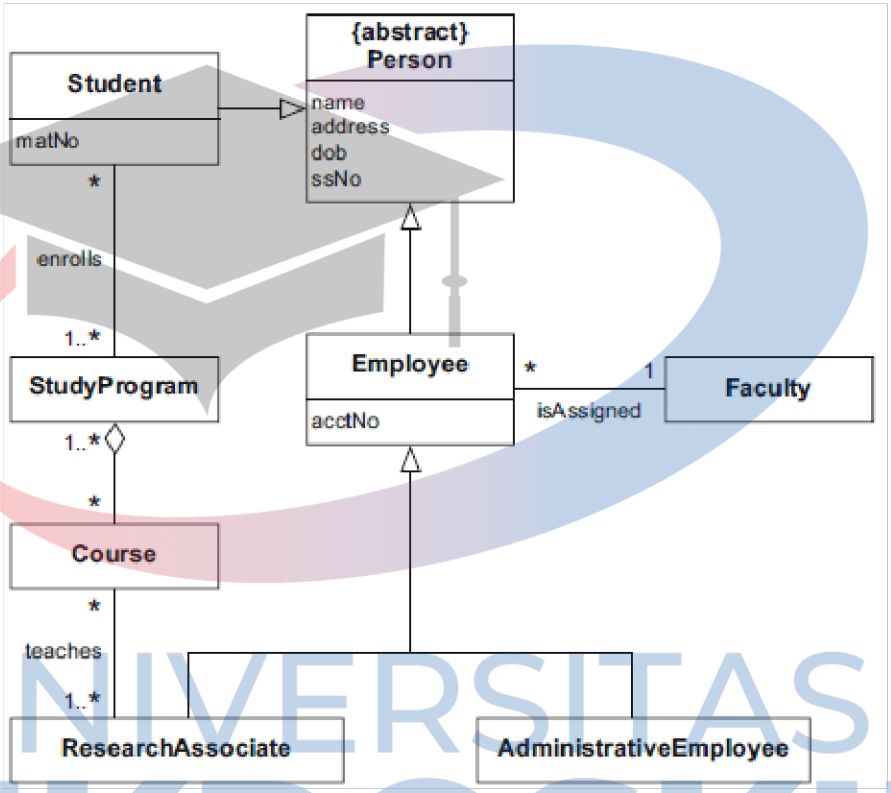
4. *Composition*

Composition menyatakan bahwa bagian tertentu hanya dapat terdapat di paling banyak satu objek komposit pada satu titik waktu tertentu. Ini menghasilkan kelipatan maksimum 1 pada ujung penjumlahan. Oleh karena itu, objek gabungan membentuk hutan pepohonan, yang menunjukkan adanya ketergantungan antara objek gabungan dan bagian-bagiannya; jika objek komposit dihapus, bagian-bagiannya juga dihapus.[15]

Gambar 2. 8. Contoh *Composition*

5. *Generalizations*

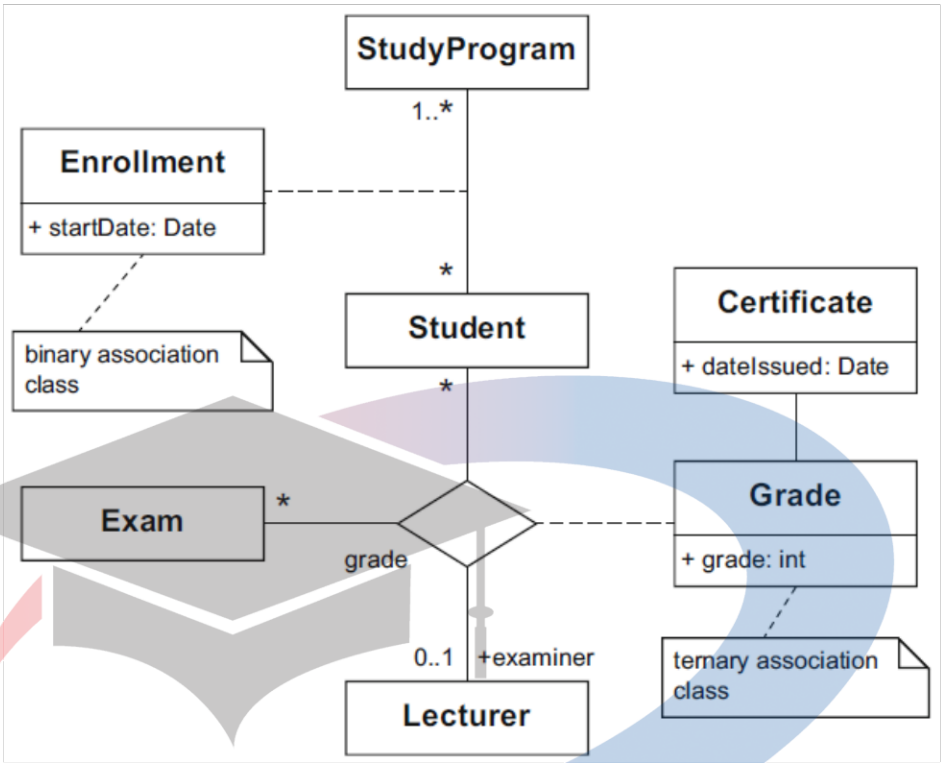
Kelas yang berbeda seringkali memiliki karakteristik yang sama. *Generalizations* dapat digunakan untuk menyoroti kesamaan antar kelas tersebut, artinya kita tidak lagi harus mendefinisikan karakteristik umum ini berkali-kali. Sebaliknya, kita dapat menggunakan generalisasi untuk mendapatkan kelas yang lebih spesifik dari kelas yang sudah ada.[15]



Gambar 2. 9. *Class Diagram With Generalizations*

6. *Class Associations*

Class associations dapat digunakan untuk menetapkan atribut atau operasi ke hubungan antara satu atau beberapa kelas daripada ke kelas itu sendiri. *Class associations* diwakili oleh kelas dan asosiasi yang dihubungkan dengan garis putus-putus. Pengaitannya bisa biner atau n-ary. Meskipun representasi mencakup banyak komponen, *class associations* adalah satu konstruksi bahasa yang memiliki properti kelas dan properti asosiasi. Oleh karena itu, dalam diagram, kelas dan asosiasi kelas asosiasi harus memiliki nama yang sama, meskipun Anda tidak harus menyebutkan keduanya.[15]



Gambar 2. 10. *Associations Class*

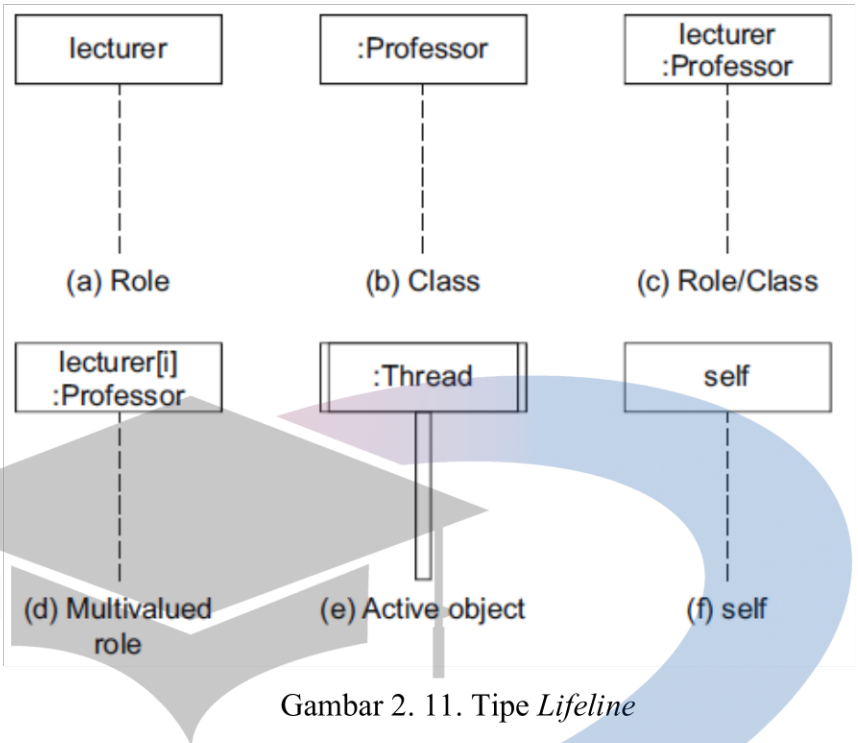
2.4.3. Sequence Diagram

Sequence diagram adalah diagram yang merepresentasikan relasi antar objek berdasarkan urutan kronologi dari pengolahan data yang terjadi antara objek untuk menyelesaikan suatu tugas. *Sequence diagram* menyediakan beragam jenis konstruksi untuk mengatur urutan kronologis data serta konsep sebagai modularisasi yang membantu memodelkan interaksi yang kompleks. [15]

Interaksi menentukan bagaimana pesan dan data dipertukarkan antara mitra interaksi. Mitra interaksi dapat berupa manusia, mitra interaksi seperti dosen atau mahasiswa, atau bukan manusia, seperti server, printer, atau perangkat lunak yang dapat dijalankan. Interaksi dapat berupa percakapan antara banyak orang - misalnya, ujian lisan. [15]

2.4.3.1. Interaction Partners

Dalam *sequence diagram*, mitra interaksi direpresentasikan dalam bentuk garis vertikal, pada umumnya garis tersebut putus-putus yang mewakili *lifetime* objek yang terkait dengannya. [15]

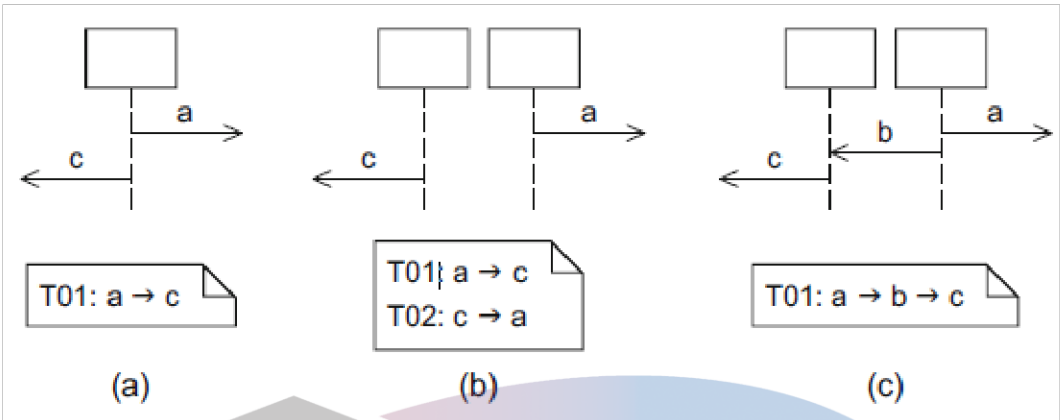


Gambar 2. 11. Tipe *Lifeline*

Pada Gambar 2.11 terlihat di ujung atas garis terdapat kepala dari *lifeline*, sebuah persegi panjang yang berisi ekspresi dalam bentuk *roleName: Class*. Ekspresi ini menunjukkan *roleName* dan kelas dari objek yang terhubung dengan *lifeline*. [15]

2.4.3.2. Exchanging Messages

Interaction partners yang terlibat dalam interaksi digambarkan pada garis horizontal dan harus diorganisir dengan urutan yang jelas. Sedangkan garis vertikal menggambarkan urutan kronologis interaksi. Dalam *sequence diagram*, interaksi menjelaskan urutan kronologis secara spesifik. Spesifikasi peristiwa melingkupi pengiriman dan penerimaan pesan atau terjadinya peristiwa berbasis waktu seperti titik waktu. Urutan di beberapa *lifeline* hanya digunakan jika terjadi pertukaran data di antara *lifeline* yang berbeda. Hubungan kronologis antara pesan a dan pesan b dinyatakan dengan simbol →.[15]

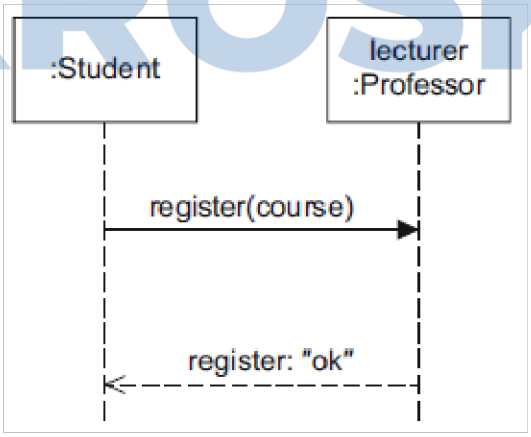


Gambar 2. 12. *Sequences of Messages and Possible Trace*

Sebagai contoh, Pada Gambar 2.12 (a), pesan a harus selalu terjadi sebelum pesan c, karena peristiwa pengiriman a terjadi sebelum peristiwa pengiriman c. Jika dua pesan tidak memiliki mitra interaksi yang sama, urutan pesan ini tidak ditentukan. Pada Gambar 2.12 (b), ini adalah kasus untuk pesan a dan c. Oleh karena itu, ada dua kemungkinan jejak: a → c dan c → a. Jika pesan b disisipkan antara a dan c dan pesan ini memaksa a dan c ke dalam urutan kronologis, satu-satunya kemungkinan jejak adalah a → b → c (lihat Gambar 2.12 (c)). [15]

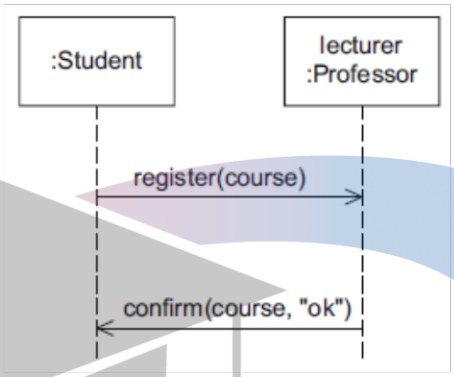
2.4.3.3. Messages

Messages direpresentasikan dalam bentuk panah dari pengirim ke penerima. Bentuk panah menjelaskan jenis komunikasi yang terjadi. *Synchronous message* digambarkan dalam bentuk panah dengan garis kontinu dan kepala panah segitiga yang berisi. [15]



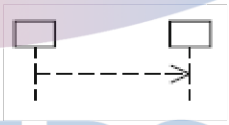
Gambar 2. 13. *Synchronous Message*

Asynchronous message berupa panah dengan garis kontinu dan kepala panah terbuka. Dalam kasus pesan sinkron, pengirim menunggu hingga menerima pesan tanggapan sebelum melanjutkan. Dalam komunikasi asinkron, pengirim melanjutkan setelah mengirim pesan ditunjukkan pada Gambar 2.14. [15]



Gambar 2. 14. *Asynchronous Message*

Response messages digambarkan dalam bentuk garis putus-putus dengan panah terbuka. Jika isi pesan tanggapan dan titik di mana pesan tanggapan dikirim dan diterima jelas dari konteksnya, maka pesan tanggapan dapat dihilangkan dalam diagram. [15]



Gambar 2. 15. *Response Message*

2.4.4. Activity Diagram

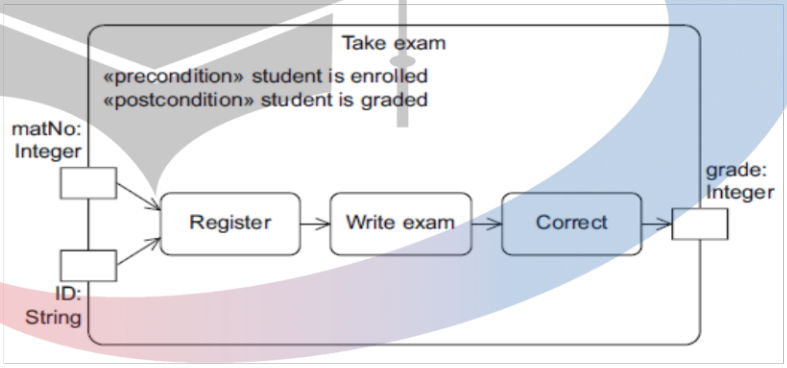
Activity diagram merupakan pemodelan yang berfokus aspek proses prosedural dari suatu sistem. Hal ini menentukan aliran kontrol dan aliran data antara berbagai langkah atau tindakan yang diperlukan untuk mengimplementasikan aktivitas. Salah satu kelebihan diagram aktivitas adalah diagram ini tidak hanya mendukung pemodelan sistem berorientasi objek, namun juga mendukung sistem non-berorientasi objek. Diagram ini membantu dalam menentukan aktivitas secara independen dari objek, yang berarti. [15]

2.4.4.1. Activities

Aktivitas merepresentasikan implementasi kasus penggunaan. Aktivitas juga dapat menentukan perilaku operasi dalam bentuk instruksi individu pada tingkat yang,

atau pada tingkat yang kurang rinci, memodelkan fungsi proses bisnis. Sebuah aktivitas direpresentasikan dalam bentuk persegi panjang yang bersudut melengkung, dan seperti halnya sebuah operasi, memiliki parameter. [15]

Untuk membuat diagram lebih mudah dibaca, penempatkan parameter input harus berada di batas kiri atau atas dan parameter output di batas kanan atau bawah aktivitas. Dengan begitu aktivitas tersebut dapat dibaca dari kiri ke kanan atau dari atas ke bawah. Nilai yang ditransfer ke aktivitas melalui parameter input tersedia untuk tindakan yang terhubung ke parameter input oleh tepi terarah. Dengan cara yang sama, parameter *output* menerima nilainya melalui tepi terarah dari tindakan dalam aktivitas. [15]



Gambar 2. 16. Contoh *Activity*

2.4.4.2. Actions

Elemen dasar dari *activity* adalah *actions*. Sebuah aksi juga digambarkan dalam bentuk persegi panjang dengan sudut melengkung, dan nama aksi tersebut berada di tengah persegi panjang tersebut. Aksi dapat digunakan untuk menentukan perilaku apa pun yang ditentukan pengguna. Tidak ada persyaratan bahasa khusus untuk deskripsi aksi. Oleh karena itu, dalam menentukan tindakan dapat menggunakan bahasa natural atau dalam bahasa pemrograman apa pun. [15]

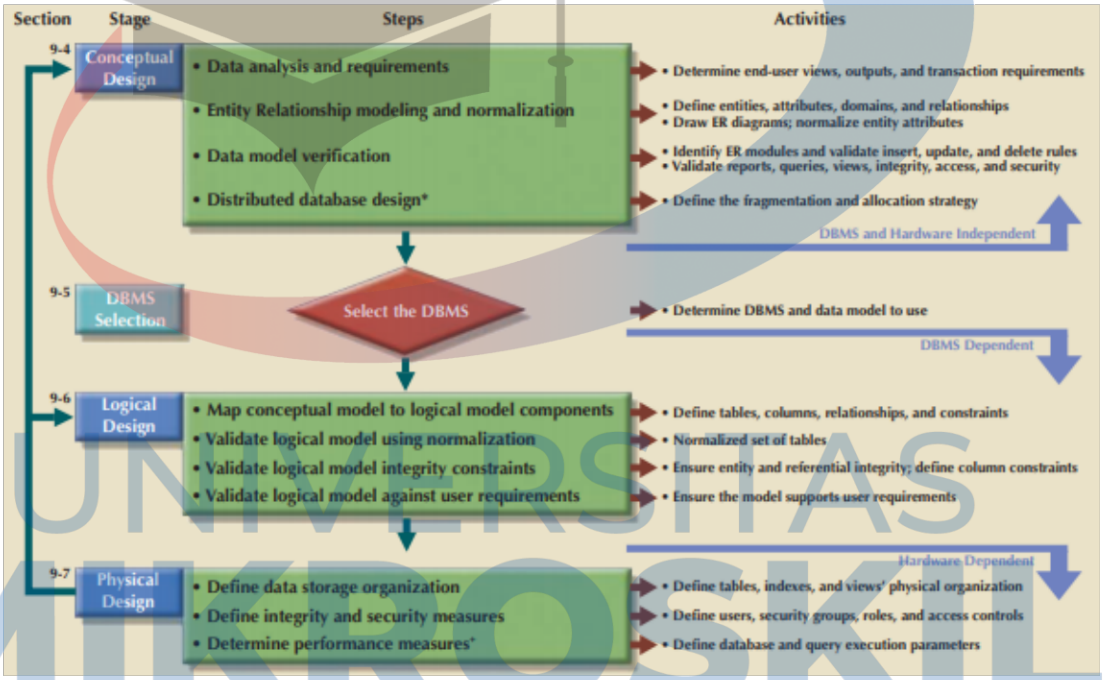
Aksi mengelola nilai input untuk menghasilkan nilai output, dan mampu melakukan penghitungan. Aksi juga dapat mengambil data dari memori dan dapat mengubah status sistem saat ini. [15]

2.5. Perancangan *Database*

Database atau basis data dapat digunakan untuk menyimpan produk, detail pelanggan, catatan anggota masyarakat atau klub, dan banyak lagi. Mereka dapat

menyimpan nama, kata sandi, alamat, alamat *email*, tanggal pendaftaran, *entri blog*, dan nomor telepon. *Database* dapat dikatakan sebagai *folder* yang berisi tabel data. Tabel data memiliki kolom dan baris; baris dalam tabel *database* disebut *record*.[18] *Database* merupakan cara yang paling efektif sebagai penyimpanan dan pengolahan data.[19]

Perancangan basis data adalah kegiatan merancang struktur basis data sesuai kebutuhan pengguna akhir, yang akan menjadi media penyimpanan dan pengelolaan data pada suatu sistem. Struktur *database* yang memenuhi semua kebutuhan pengguna harus dirancang dengan hati-hati. Perancangan *database* menjadi aspek penting ketika bekerja menggunakan *database*. [19]



Gambar 2. 17. Proses Perancangan *Database*

Perancangan *database* terbagi menjadi 4 tahapan, antara lain[19]:

1. Membuat desain konseptual

   Tahapan ini menggunakan teknik pemodelan data dengan tujuan menghasilkan model struktur *database* yang mewakili objek dunia nyata. Hasil dari tahapan ini adalah model data konseptual yang merepresentasikan entitas data utama, atribut, hubungan, dan batasan dari domain masalah yang diberikan. Desain ini biasanya mencakup representasi grafis serta teks keterangan dari elemen data utama, hubungan, dan batasan.

Desain konseptual memiliki 4 tahapan, antara lain[19]:

a. Analisis dan persyaratan data
b. Pemodelan dan normalisasi hubungan entitas
c. Verifikasi model data
d. Desain database terdistribusi

2. Pemilihan perangkat lunak DBMS

Ada beberapa faktor paling umum dalam pemilihan perangkat lunak DBMS, antara lain[19]:

a. Biaya
b. Fitur dan alat yang disediakan DBMS
c. Model yang mendasari, seperti hierarki, jaringan, relasional, objek / relasional, atau *object-oriented*.
d. Portabilitas, apakah DBMS dapat menjadi portabel lintas platform, sistem, dan bahasa.
e. Persyaratan perangkat keras DBMS.

3. Membuat desain logis

Tahapan ini merancang keseluruhan *database* berdasarkan model data tertentu tetapi terlepas dari detail tingkat fisik. Desain logis harus memetakan semua objek dalam model konseptual ke dalam konstruksi spesifik yang digunakan oleh model *database* yang dipilih.

Desain logis memiliki 4 tahapan, antara lain[19]:

a. Memetakan model konseptual ke komponen model logis.
b. Memvalidasi model logis menggunakan normalisasi.
c. Memvalidasi batasan integritas model logis.
d. Memvalidasi model logis terhadap persyaratan pengguna.

4. Membuat desain fisik

Tahapan ini menentukan organisasi penyimpanan data dan karakteristik akses data dari *database* untuk memastikan integritas, keamanan, dan kinerjanya. Karakteristik penyimpanan adalah fungsi dari jenis perangkat yang didukung oleh perangkat keras, jenis metode akses data yang didukung oleh sistem, dan DBMS.

2.6. *Usability Testing*

Usability testing merupakan jangkauan suatu sistem yang digunakan oleh pengguna tertentu untuk mencapai tujuan dengan efektivitas, efisiensi dan kepuasan. Ada dua catatan yang perlu diperhatikan dalam *usability testing* ini, antara lain[20]:

1. Pengguna, tujuan, dan konteks penggunaan "tertentu" maksudnya adalah kombinasi tertentu dari pengguna, tujuan, dan konteks penggunaan yang dipertimbangkan kegunaannya.
2. Kata "kegunaan" juga digunakan sebagai kualifikasi untuk merujuk pada pengetahuan desain, kompetensi, aktivitas, dan atribut desain yang berkontribusi pada kegunaan, seperti keahlian kegunaan, profesional kegunaan, teknik kegunaan, metode kegunaan, evaluasi kegunaan, heuristik kegunaan.

Efektivitas adalah tingkat keakuratan dan kelengkapan yang diperoleh pengguna untuk mencapai tujuan pengguna.[20] Efisiensi merupakan tingkat kecepatan pengguna dalam mempelajari hasil perancangan.[21] Kepuasan merupakan tingkat kepuasan pengguna dalam terhadap rancangan yang disediakan. [21]