

## **BAB 2**

### **LANDASAN TEORI**

#### **2.1 Sistem**

Suatu sistem adalah jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran tertentu. ( Kristanto, 2007).

Sistem adalah sebuah tatanan (keterpaduan) yang terdiri atas sejumlah komponen fungsional (dengan satuan fungsi / tugas khusus) yang saling berhubungan dan secara bersama-sama bertujuan untuk memenuhi suatu proses / pekerjaan tertentu. (Fathansyah, 2002).

Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan , berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran yang tertentu (Jogiyanto, 2005).

#### **2.2 Informasi**

Informasi juga berarti kumpulan data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya. (Kristanto, 2007).

Informasi adalah rangkaian data yang mempunyai sifat sementara, tergantung dengan waktu, mampu memberi kejutan atau surprise pada yang menerimanya. Informasi dapat juga dikatakan sebagai data yang telah diproses, yang mempunyai nilai tentang tindakan atau keputusan (Witarto, 2004).

Informasi diartikan sebagai data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya (Jogiyanto,2005).

#### **2.3 Sistem Informasi**

Sistem informasi adalah sistem yang diciptakan oleh para analisis dan manajer guna melaksanakan tugas khusus tertentu yang sangat esensial bagi berfungsinya organisasi (George M.Scott, 2001).

Sistem informasi adalah suatu sistem didalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan (Jogiyanto, 2005).

## **2.4 Data**

Data merupakan fakta atau bagian dari fakta yang mengandung arti, yang dihubungkan dengan kenyataan, gambar-gambar, kata-kata, angka-angka, huruf atau simbol-simbol menyatakan suatu ide objek kondisi atau situasi dan lain-lain (Fatansyah, 2001).

Data adalah kenyataan yang menggambarkan suatu kejadian-kejadian dan kesatuan nyata. Kejadian adalah sesuatu yang terjadi pada saat tertentu, contohnya transaksi. Kesatuan nyata adalah berupa suatu objek nyata seperti tempat, benda dan orang yang betul-betul ada dan terjadi (Kristanto, 2007).

Data Merupakan kenyataan yang menggambarkan suatu kejadian-kejadian dan kesatuan nyata (Jogiyanto, 2008).

## **2.5 Raport**

*Raport* adalah sebuah buku yang berisi nilai dari setiap mata pelajaran yang diperoleh dari penghitungan nilai ulangan harian, ulangan tengah semester dan ulangan akhir semester.

## **2.6 Metode perancangan**

### ***2.6.1 System Development Life Cycle***

*System Development Life Cycle* (SDLC) adalah suatu pengembangan sistem yang mengidentifikasi semua kegiatan yang diperlukan untuk membangun, meluncurkan dan memelihara sistem informasi. Biasanya SDLC mencakup semua kegiatan yang merupakan bagian dari analisis sistem, desain sistem, pemrograman, pengujian, dan pemeliharaan sistem serta proses manajemen proyek lainnya yang diperlukan untuk keberhasilan penyebaran sistem informasi (Satzinger, Jackson, & Burd, 2012). Terdapat 6 proses pokok yang dibutuhkan untuk mengembangkan aplikasi baru, yaitu:

1. Identifikasi masalah atau kebutuhan yang diperlukan dan mendapatkan persetujuan untuk beralih ke proses selanjutnya.
2. Merencanakan dan memonitor proyek apa saja yang harus dilakukan, bagaimana melakukannya dan siapa yang melakukannya.
3. Menemukan dan memahami detail dari masalah atau kebutuhan dari sistem.
4. Mendesain komponen sistem untuk memecahkan masalah dan menemukan solusi berdasarkan kebutuhan.
5. Membangun, mengetes dan mengintegrasikan komponen sistem.
6. Menyelesaikan tes sistem dan menyebarkan atau menginformasikan solusi kepada user.

#### 1.6.1.1 SDLC Model Air Terjun (Waterfall)

Salah satu model SDLC yang paling awal dan paling banyak digunakan adalah SDLC model air terjun (waterfall). Pemodelan ini menggambarkan pendekatan sekuensial beberapa tahap yang biasanya disebut juga dengan model *waterfall*. Model waterfall menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau urut dimulai dari analisis, desain pengkodean, pengujian dan tahap support (Satzinger, Jackson, & Burd, 2012). Dalam waterfall model diperlukan 6 tahapan, yaitu :

##### 1. *Project initiation*

*Project initiation* adalah proses fundamental untuk memahami proses pembangunan sistem informasi yg harus dibangun, dan menentukan bagaimana suatu tim proyek akan membangunnya.

##### 2. *Project planning*

Pada fase ini tim proyek melakukan investigasi terhadap kondisi sistem yang ada, kesempatan pengembangan, dan pengembangan konsep pada sistem yang baru. Fase ini membahas mengenai keputusan strategis dalam pengembangan sistem.

### 3. *Analysis*

*Analysis phase* adalah menganalisa workflow sistem informasi yang sedang berjalan dan mengidentifikasi apakah workflow telah efisien dan sesuai standar tertentu.

### 4. *Design*

*Design phase* menentukan bagaimana sistem akan dioperasikan dalam hardware, software, dan infrastruktur jaringan yang akan ditempatkan, tampilan form, laporan yang akan digunakan, spesifikasi program, database, dan dokumen yang akan dibutuhkan.

### 5. *Implementation*

Fase ini membahas tentang bagaimana sistem dibangun. Fase ini yang menjadi fokus utama dari sebagian besar proses pengembangan sistem informasi. Karena fase ini merupakan fase yang terpanjang dan termahal untuk setiap bagian pengembangan.

### 6. *Deployment*

Tahapan ini merupakan tahap pemeliharaan sistem melalui evaluasi dan perbaikan berkala.

## **2.6.2 *Object-Oriented Analysis and Design***

*Object-Oriented Analysis and Design with the Unified Process* (OOAD) menurut (Satzinger, Jackson, & Burd, 2012) adalah:

- *Object Oriented Programming (OOP)* menuliskan tentang pernyataan dalam bahasa pemrograman untuk mendefinisikan tipe dari masing-masing objek.
- *Object-Oriented Analysis (OOA)* adalah semua jenis objek yang melakukan pekerjaan dalam sistem dan menunjukkan interaksi pengguna apa yang dibutuhkan untuk menyelesaikan tugas-tugas. Objek diartikan sebagai suatu

hal dalam sistem komputer yang dapat merespon pesan-pesan.

*Object-Oriented Design (OOD)* adalah semua jenis objek yang diperlukan untuk berkomunikasi dengan orang dan perangkat dalam sistem, menunjukkan bagaimana objek berinteraksi untuk menyelesaikan tugas, dan menyempurnakan definisi dari masing-masing jenis objek sehingga dapat diimplementasikan dengan bahasa tertentu.

### **2.6.3 Unified Process**

*Unified process (UP)* adalah metodologi pengembangan sistem berbasis objek (Satzinger, Jackson, & Burd, 2012). Metode ini sudah menjadi salah satu metode yang banyak digunakan dalam pengembangan sistem berorientasi objek. UP memperkenalkan pendekatan baru untuk siklus hidup pengembangan sistem yang menggabungkan perulangan (*iterations*) dan tahapan (*phases*) yang disebut dengan siklus hidup UP (*UP life cycle*). UP mendefinisikan empat tahapan siklus hidup yaitu *inception*, *elaboration*, *construction*, dan *transition*.

#### **2.6.3.1 Langkah–Langkah Unified Process (UP)**

Langkah-langkah *unified process (UP)* menurut (Satzinger, Jackson, & Burd, 2012) adalah sebagai berikut :

##### *1. Inception phase*

Seperti di dalam setiap tahap perencanaan proyek, fase awal dimulai dari seorang manajer proyek mengembangkan dan menyempurnakan visi untuk sistem baru, menunjukkan bagaimana hal tersebut akan meningkatkan operasi dan memecahkan masalah yang ada. Pada dasarnya, manajer proyek akan membuat kasus bisnis untuk sistem baru, membuktikan bahwa manfaat sistem baru akan lebih besar daripada biaya pembangunan (*construction*). Ruang lingkup sistem juga harus

didefinisikan sehingga jelas apakah proyek ini akan berhasil dicapai atau tidak. Mendefinisikan ruang lingkup meliputi identifikasi semua persyaratan utama untuk sistem. Tahap awal biasanya diselesaikan dalam satu iterasi, dan di dalam iterasi tersebut, bagian dari sistem yang sebenarnya dapat dirancang, dilaksanakan dan diuji. Sebagai perangkat lunak yang dikembangkan, anggota tim harus mengkonfirmasi bahwa visi system masih sesuai harapan pengguna.

### *2. Elaboration phase*

Fase elaborasi biasanya melibatkan beberapa iterasi, dan iterasi awal biasanya menyelesaikan identifikasi dan definisi dari semua persyaratansistem. Karena UP adalah pendekatan adaptif untuk pembangunan, persyaratan diharapkan berkembang dan berubah setelah dimulainya proyek. Tahapan iterasi pada elaborasi juga melengkapi analisis, desain, dan pelaksanaan arsitektur inti sistem. Biasanya, aspek dari sistem yang menimbulkan resiko terbesar diidentifikasi dan dilaksanakan terlebih dahulu sampai pengembang mengetahui persis bagaimana aspek tertinggi resiko proyek akan bekerja. Pada akhir fase elaborasi, manajer proyek harus memiliki perkiraan yang lebih realistis untuk biaya proyek dan jadwal, dan kasus bisnis atas proyek dapat dikonfirmasi terlebih dahulu. Salah satu tujuan utama dari fase elaborasi adalah untuk melakukan penelitian yang diperlukan data atau fakta sehingga semua kebutuhan pengguna diidentifikasi secara jelas dan rinci.

### *3. Construction phase*

Tahap konstruksi melibatkan beberapa iterasi yang meneruskan atau melanjutkan desain dan implementasi sistem. Arsitektur inti dan aspek tertinggi resiko sistem sudah selesai pada tahap ini. Fokus utama di dalam tahap ini adalah bagaimana merinci sistem kontrol, seperti validasi data, *fine-tuning* antar muka pengguna desain, menyelesaikan fungsi pemeliharaan data rutin, dan menyelesaikan bantuan serta preferensi penggunaan fungsi.

#### 4. *Transition phase*

Selama fase transisi atau tahap akhir dari UP, satu atau lebih iterasi akhir yang melibatkan penerimaan pengguna (*end users*), beta tes akhir, dan sistem dibuat siap untuk dioperasikan. Setelah sistem ini beroperasi, maka akan perlu didukung dan dipertahankan fungsi kegunaan dari sistem tersebut.

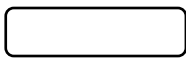

#### 2.6.4 *Unified Model Language*

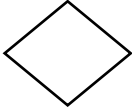



*Unified Modeling Language* (UML) adalah standar bahasa pemodelan konstruksi dan notasi *object oriented* yang didefinisikan oleh kelompok manajemen obyek (Satzinger, Jackson dan Burd, 2012). Unified model language (UML) terbagi menjadi beberapa model diagram yang biasanya digunakan, yaitu :

##### 1. *Activity Diagram*

*Activity diagram* adalah jenis diagram alur kerja yang menggambarkan kegiatan pengguna atau sistem dan aliran sekuensial kegiatan mereka (Satzinger, Jackson dan Burd, 2012). Bentuk oval dalam *activity Diagram* menunjukkan kegiatan yang dilakukan dalam transaksi. Tanda panah menunjukkan hubungan kegiatan secara berurutan. Lingkaran hitam menjelaskan awal kegiatan hingga akhir kegiatan. Simbol diamond menjelaskan poin aliran proses dari hasil eksekusi. Berikut simbol-simbol yang terdapat pada *activity diagram* ditunjukkan pada **tabel 2.1** dan contoh dari *Activity diagram* ditunjukkan pada **gambar 2.1**.

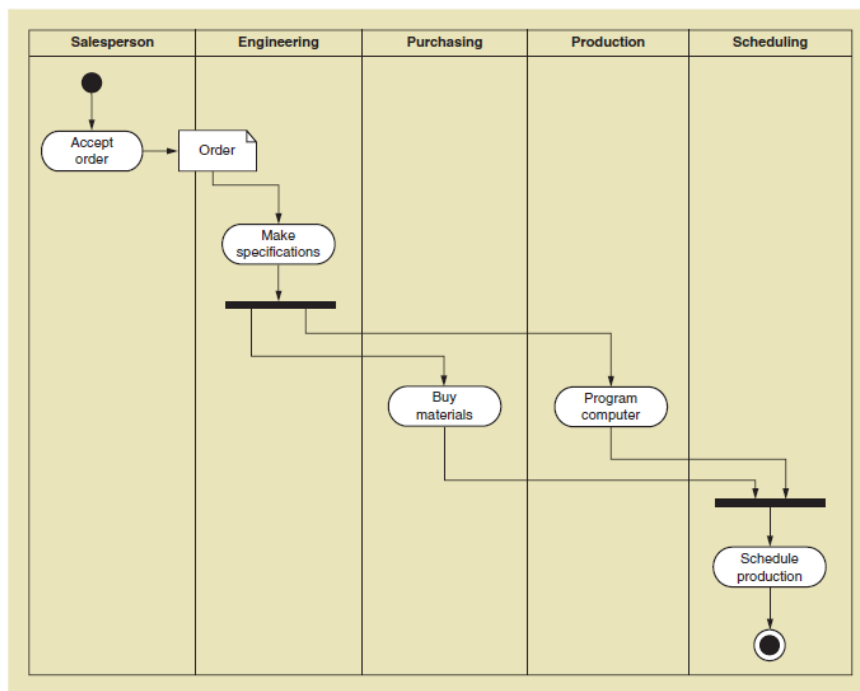
**Tabel 2.1 Simbol Activity Diagram (Satzinger, Jackson dan Burd, 2012)**

	Simbol	Gambar	Keterangan
1	<i>Rounded rectangle</i>		Bentuk untuk fungsi sistem yang spesifik.
2	<i>Arrows</i>		Menunjukkan aliran kerja pada sistem.

3	<i>Decision activity</i>		Sebagai bentuk kerja yang memiliki cabang, bisa memiliki lebih dari satu bentuk kerja selanjutnya.
4	<i>Horizontal line</i>		Menjelaskan aktivitas paralel yang sedang terjadi.
5	<i>Start node</i>		Menunjukkan awal dari proses yang akan berjalan.
6	<i>End node</i>		Menunjukkan akhir dari suatu proses yang berjalan.

Berikut adalah contoh activity diagram atau diagram aktivitas untuk kasus penggunaan item kapal. Salah satu kekuatan diagram aktivitas adalah memberikan gambaran yang lebih grafis tentang alur kegiatan. Diagram tersebut mengilustrasikan kedua rangkaian langkah yang berulang, yaitu setiap *sale item* dalam penjualan, dan titik keputusan untuk memilih set langkah mana yang harus dilakukan.





**Gambar 2.1** Contoh *Activity Diagram* (Satzinger, Jackson dan Burd, 2012)

### 1. *Use Case Diagram*

*Use Case Diagram* adalah model UML yang digunakan untuk mendokumentasikan bermacam-macam peran user dan cara mereka berinteraksi dengan system (Satzinger, Jackson, & Burd, 2012). Dalam analisis *Use Case*, kita mengidentifikasi suatu entitas yang disebut *actor*, yang berhubungan dengan sistem. Tujuan dari *Diagram Use Case* adalah untuk mengidentifikasi bagaimana sistem akan digunakan oleh *user*. Berikut ini adalah komponen – komponen dari *Use Case Diagram* :

- ***System Boundary***

menggambarkan batasan antar sistem (*use case*) dengan *actor*.

- ***Actor***

Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan – pekerjaan tertentu.

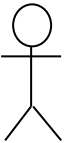
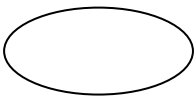


- ***Use Case***

Menggambarkan perilaku interaksi antara *actor* dengan *system*.

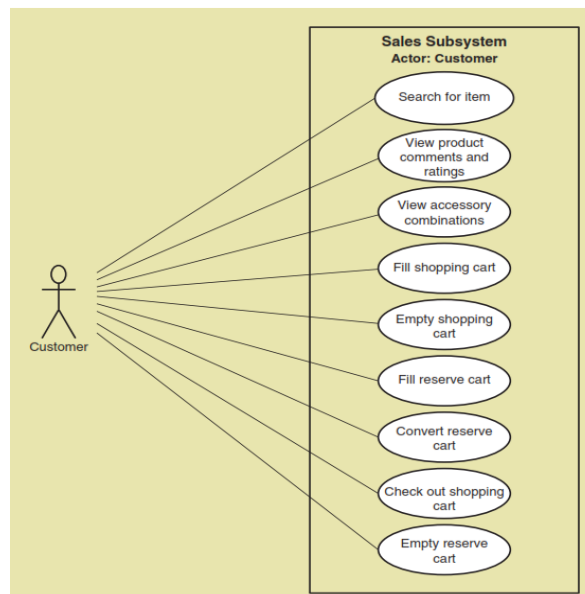
- **Communication**  
Menggambarkan hubungan antara *use case* dengan *actor*.

Berikut adalah simbol-simbol use case diagram ditunjukkan pada tabel 2.2 dan contoh use case diagram ditunjukkan pada gambar 2.2.

**Tabel 2.2 Simbol Use Case Diagram (Satzinger, Jackson dan Burd, 2012)**

No	Simbol	Gambar	Keterangan
1	Aktor		Aktor adalah segala sesuatu yang perlu berinteraksi dengan sistem untuk pertukaran informasi. Aktor dapat berupa manusia, organisasi, sistem informasi yang lain, dan perangkat eksternal. Label dari peran yang dilakukan aktor diletakkan di atas simbol.
2	<i>Use case</i>		<i>Use case</i> adalah fungsi dalam sistem yang merepresentasikan tujuan dari sistem dan mendeskripsikan aktivitas dan interaksi user dengan sistem untuk mencapai tujuan tersebut.
3	Rectangular		Menjadi tempat untuk menaruh setiap kerja yang dilakukan.
4	Association		Association adalah abstraksi dari penghubung antara aktor dan use case

Berikut ini merupakan contoh dari *use case* diagram, dimana dapat dilihat contoh dari *use case* diagram yang menjelaskan *use case* yang ditampilkan untuk *customer* dalam membeli suatu barang.



**Gambar 2.2 Contoh Use Case Diagram (Satzinger, Jackson dan Burd, 2012)**

### 1. Use case description

*Use case description* merupakan penjelasan terperinci mengenai proses dari suatu *use case* atau bisa disebut juga sebagai daftar kasus penggunaan diagram *use case* yang memberikan gambaran dari semua penggunaan kasus untuk sistem (Satzinger, Jackson, dan Burd, 2012). Informasi rinci tentang setiap kasus penggunaan digambarkan dengan menggunakan deskripsi kasus. *Use case description* dibedakan menjadi tiga, yaitu :

#### a. Brief Description

*Brief Description* dapat digunakan untuk *use case* yang sederhana, khususnya sistem yang dikembangkan untuk aplikasi kecil yang mudah dimengerti. Berikut adalah contoh *brief description* yang terdapat pada gambar 2.3.

Berikut ini contoh *brief description* untuk *use case* membuat akun pelanggan, mencari pelanggan, dan penyesuaian proses.

Use case	Brief use case description
<i>Create customer account</i>	User/actor enters new customer account data, and the system assigns account number, creates a customer record, and creates an account record.
<i>Look up customer</i>	User/actor enters customer account number, and the system retrieves and displays customer and account data.
<i>Process account adjustment</i>	User/actor enters order number, and the system retrieves customer and order data; actor enters adjustment amount, and the system creates a transaction record for the adjustment.

**Gambar 2.3 Contoh *Brief Description* (Satzinger, Jackson, & Burd, 2012).**

## 2. *Fully Developed Description*

*Fully Developed Description* adalah metode paling formal untuk mendokumentasikan *use case* karena menjelaskan secara rinci setiap proses dalam *use case diagram*. Berikut adalah contoh *Fully Developed Description* yang terdapat pada gambar 2.4.

Berikut merupakan contoh *fully developed description* dengan nama *use case* membuat akun pelanggan.

<b>Use case name:</b>	<i>Create customer account.</i>	
<b>Scenario:</b>	Create online customer account.	
<b>Triggering event:</b>	New customer wants to set up account online.	
<b>Brief description:</b>	Online customer creates customer account by entering basic information and then following up with one or more addresses and a credit or debit card.	
<b>Actors:</b>	Customer.	
<b>Related use cases:</b>	Might be invoked by the <i>Check out shopping cart</i> use case.	
<b>Stakeholders:</b>	Accounting, Marketing, Sales.	
<b>Preconditions:</b>	Customer Account subsystem must be available. Credit/debit authorization services must be available.	
<b>Postconditions:</b>	Customer must be created and saved. One or more Addresses must be created and saved. Credit/debit card information must be validated. Account must be created and saved. Address and Account must be associated with Customer.	
<b>Flow of activities:</b>	<b>Actor</b>	<b>System</b>
	1. Customer indicates desire to create customer account and enters basic customer information.	1.1 System creates a new customer. 1.2 System prompts for customer addresses.
	2. Customer enters one or more addresses.	2.1 System creates addresses. 2.2 System prompts for credit/debit card.
	3. Customer enters credit/debit card information.	3.1 System creates account. 3.2 System verifies authorization for credit/debit card. 3.3 System associates customer, address, and account. 3.4 System returns valid customer account details.
<b>Exception conditions:</b>	1.1 Basic customer data are incomplete. 2.1 The address isn't valid. 3.2 Credit/debit information isn't valid.	

**Gambar 2.4 Contoh *Fully Developed Description* (Satzinger, Jackson, & Burd, 2012).**

### 1. Domain Model Class Diagram

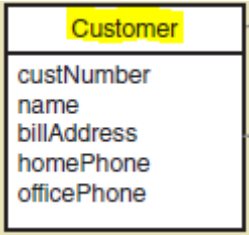
*Domain class diagram* adalah suatu *class* yang menunjukkan hubungan pada masalah pengguna domain dimana *domain class* diagram terdiri dari nama *class* dan *attribute* yang menyatakan objek-objek *class* di dalamnya (Satzinger, Jackson, & Burd, 2012).

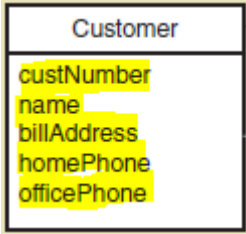

Dalam *class diagram*, juga dikenal berdasarkan karakteristik *class* yang sama dan hal tersebut berguna untuk menyusun *class* mulai dari karakteristik yang umum hingga karakteristik yang khusus. *Class* memiliki karakteristik yang umum disebut *superclass*. Sedangkan *class* yang memiliki karakteristik khusus disebut *subclass*. Sebuah *subclass* dapat memiliki karakteristiknya *superclass* dengan penurunan karakteristik atau *inheritance*.

Dalam hierarki *class diagram* terdapat dua jenis *whole-part hierarchies*, yaitu *aggregation* dan *composition*. *Aggregation* digunakan untuk menggambarkan sebuah hubungan antara agregat (keseluruhan) dan komponennya (bagian-bagian) dimana bagian-bagian tersebut dapat berdiri sendiri secara terpisah, sedangkan *composition* digunakan untuk menggambarkan hubungan keterikatan yang lebih kuat, dimana tiap-tiap bagian tidak dapat berdiri sendiri secara terpisah (Satzinger, Jackson, dan Burd, 2012).

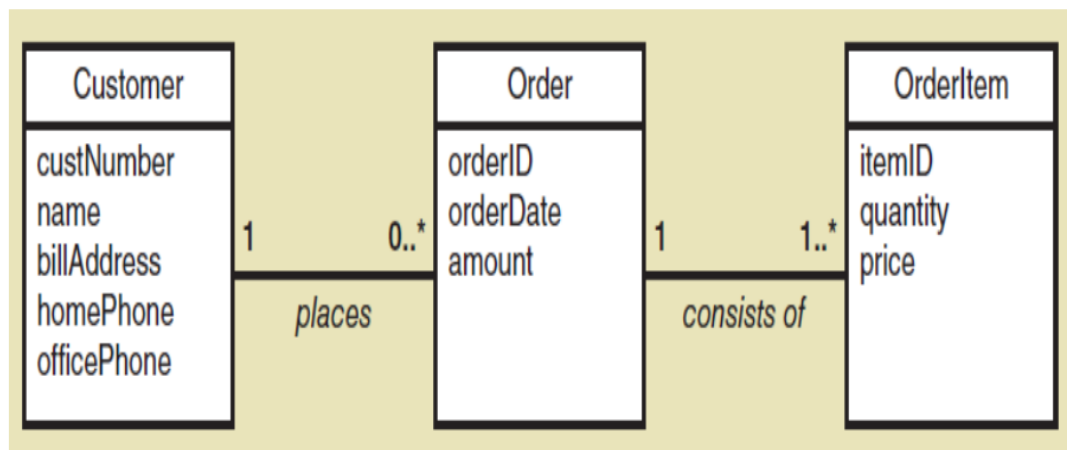
Berikut ini merupakan penjelasan untuk *multiplicity indicator*.

**Tabel 2.3 Simbol Domain Model Class Diagram (Satzinger, Jackson, & Burd, 2012)**

No	Nama	Simbol	Arti
1	Nama <i>class</i>	 <p>The diagram shows a rectangular box representing a class. The top part of the box is a yellow header containing the class name 'Customer'. Below the header, the attributes are listed: 'custNumber', 'name', 'billAddress', 'homePhone', and 'officePhone'.</p>	Mendefinisikan nama dari kelas. Dalam penulisannya menggunakan kaidah penulisan <i>camelback</i> ; yaitu setiap kata digabungkan tanpa spasi dan awal setiap kata menggunakan huruf kapital.

No	Nama	Simbol	Arti
2	Nama <i>attribute</i>		Menjabarkan atribut yang dimiliki sebuah kelas. Dalam penulisannya menggunakan kaidah penulisan camelback; yaitu setiap kata digabungkan tanpa spasi dan awal setiap kata menggunakan huruf kapital.
3	Garis asosiasi		Hubungan statis antara dua kelas bersama dengan multiplicitas/ <i>multiplicity</i> .
4	Indikator <i>multiplicity</i>	0..1	Nol atau satu
5	Indikator <i>multiplicity</i>	1	Hanya satu
6	Indikator <i>multiplicity</i>	0..*	Nol atau lebih
7	Indikator <i>multiplicity</i>	1..*	Satu atau lebih
8	Indikator <i>multiplicity</i>	N	Hanya n (dimana $n > 1$ )
9	Indikator <i>multiplicity</i>	*	Banyak
10	Indikator <i>multiplicity</i>	0..n	Nol ke n (dimana $n > 1$ )
11	Indikator <i>multiplicity</i>	..n	Satu ke n (dimana $n > 1$ )
12	Indikator <i>multiplicity</i>	..m	Dimana n dan m, keduanya $> 1$
13	Indikator <i>multiplicity</i>	..*	n atau lebih, dimana $n > 1$

**Gambar 2.5** menunjukkan contoh lain dari diagram kelas model domain, diagram ini merupakan proses pembelian barang yang dilakukan oleh customer.

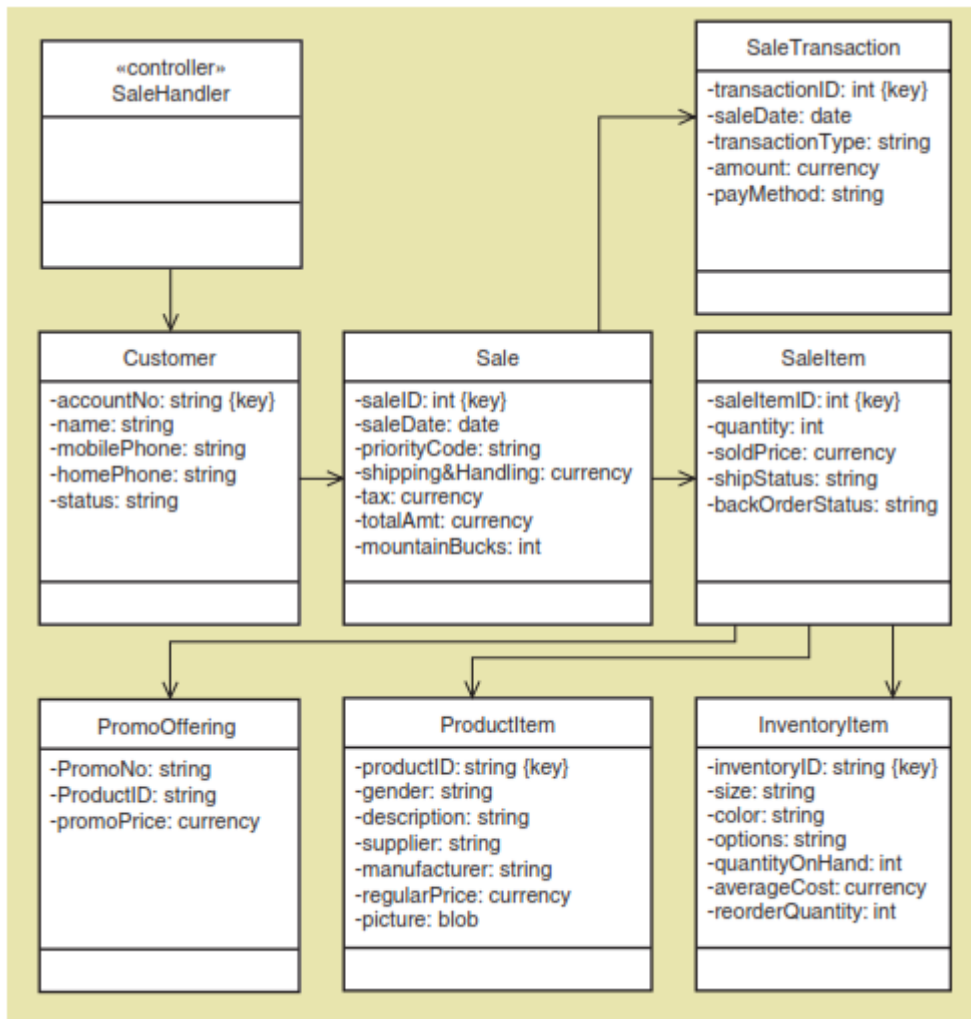


**Gambar 2.5 Contoh Domain Model Class Diagram (Satzinger, Jackson dan Burd, 2012)**

### 1.1 *First Cut Design Model Class Diagram*

First Cut Design Class Diagram dikembangkan dengan memperluas model domain class diagram dan memerlukan dua langkah yaitu mengelaborasi atribut-atribut dengan tipe dan nilai informasi inisial dan langkah ke dua adalah menambahkan panah navigasi visibilitas (Satzinger, Jackson, & Burd, 2012). Berikut adalah contoh *first cut design model diagram* yang terdapat pada gambar 2.6.

Berikut ini merupakan contoh *first cut class diagram* untuk *use case* penjualan telepon.



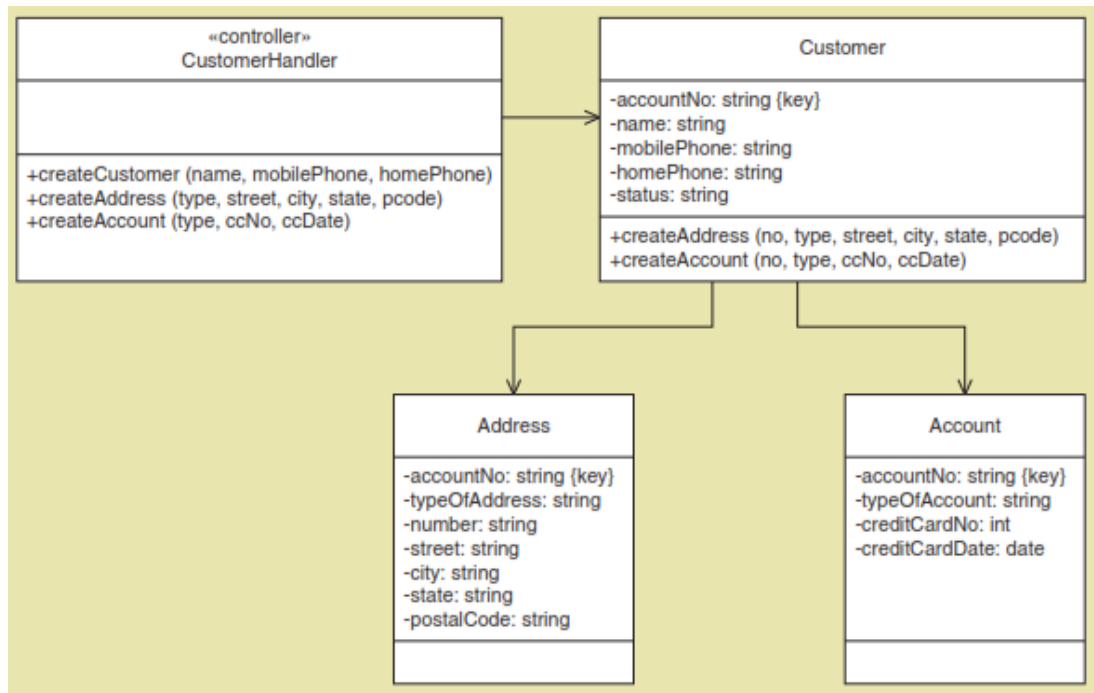
**Gambar 2.6 Contoh First Cut Design Modeling Diagram (Satzinger, Jackson, dan Burd, 2012)**

### 1.2 Update Design Class Diagram

*Updated Class Diagram* merupakan pengembangan dari *first-cut class diagram* (Satzinger, Jackson, and Burd (2005)). Pada *updated class diagram*, informasi *method* dapat ditambahkan ke dalam *class* dan dilakukan pembaharuan arah panah navigasi yang didapat dari *sequence diagram* yang telah dibuat sebelumnya. Semua handler akan menjadi *class* baru. Berikut ini adalah contoh update design class diagram yang terdapat pada gambar 2.5.

Berikut ini merupakan contoh untuk *update design class diagram*.



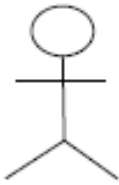
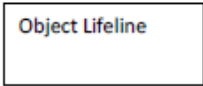






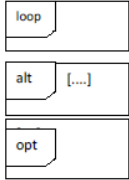
**Gambar 2.7 Contoh Update Design Class Diagram (Satzinger, Jackson dan Burd, 2012)**

## 2. System Sequence Diagram

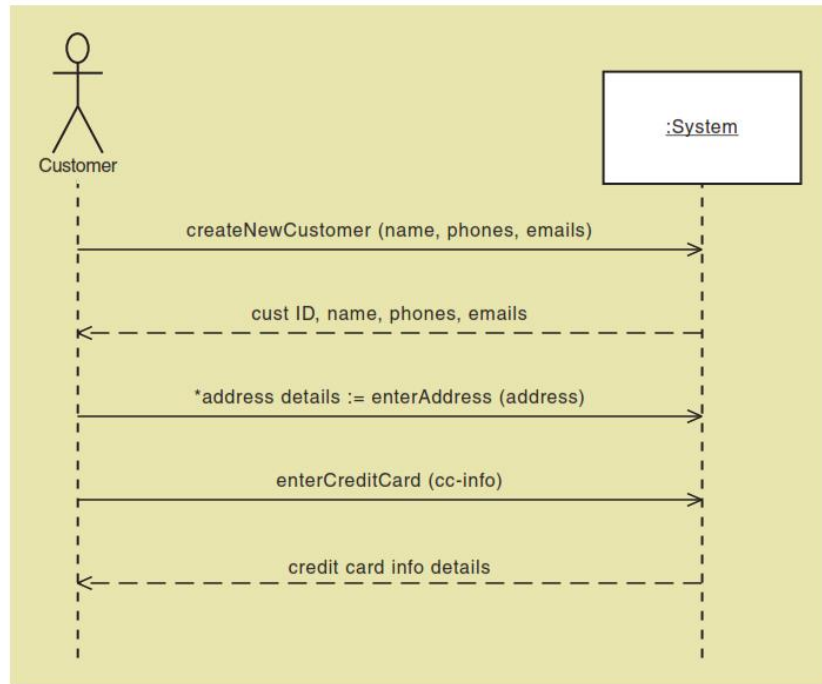
*System Sequence Diagram* adalah diagram yang menunjukkan urutan pesan antara aktor eksternal dan sistem selama *use case* atau scenario (Satzinger, Jackson dan Burd, 2012). Sequence diagram diawali dengan memilih aktivitas yang akan menyebabkan aktivitas tersebut, melakukan proses yang sesuai, dan perubahan apa yang terjadi secara input atau output. Masing-masing objek, termasuk actor, memiliki lifeline berupa garis vertical. Message digambarkan sebagai garis dengan panah dari actor ke objek atau objek ke objek atau sebaliknya. Berikut ini simbol-simbol *system sequence diagram* ditunjukkan pada tabel 2.4 dan contoh dari *system sequence diagram* ditunjukkan pada gambar 2.8.

Tabel 2.4 Simbol *System Sequence Diagram* (Satzinger, Jackson dan Burd, 2012)

	Simbol	Gambar	Keterangan
1	<i>Aktor</i>		Figur stik merepresentasikan aktor yang berinteraksi dengan sistem dimana aktor menggambarkan seseorang, pada hal ini figur stik mempunyai fungsi yang sama seperti halnya pada <i>usecase</i> .
2	<i>Object</i>		Sistem dilambangkan dengan bentuk kotak penggunaan (:) titik dua merupakan standar yang digunakan untuk menunjukkan kejadian ( <i>instance</i> ) yang sedang berjalan.
3	<i>Lifeline</i>		Riwayat atau waktu hidup ( <i>life</i> ) dari aktor atau sistem.
4	<i>Message</i>		Menjelaskan hubungan satu arah yang terjadi antara <i>user</i> ke komputer dimana merupakan hal yang dilakukan oleh <i>user</i> dan mempengaruhi sistem, atau dapat juga merupakan <i>request user</i> ke sistem.
5	<i>Return Message</i>		Bentuk panah dengan garis putus-putus menjelaskan hubungan satu arah yang

			berjalan dari komputer atau sistem menuju ke <i>user</i> , dimana biasanya hal yang dilakukan merupakan <i>feedback</i> dari sistem atas tindakan yang dilakukan oleh <i>user</i> .
6	<i>Object Active</i>		Periode waktu saat aktor atau system sedang dalam kondisi aktif (ada proses yang berjalan) pada suatu interaksi yang terjadi.
7	<i>Loop Combine Fragment</i>		Bentuk kotak dapat melambangkan fungsi <i>loop</i> , <i>alternate</i> (alt), <i>option</i> (opt). Kondisi yang ada dalam kurung siku ([ ]) merupakan kondisi yang harus dipenuhi selama proses berjalan. Bentuk kotak dapat melambangkan fungsi <i>loop</i> , <i>alternate</i> (alt), <i>option</i> (opt). Kondisi yang ada dalam kurung siku ([ ]) merupakan kondisi yang harus dipenuhi selama proses berjalan.

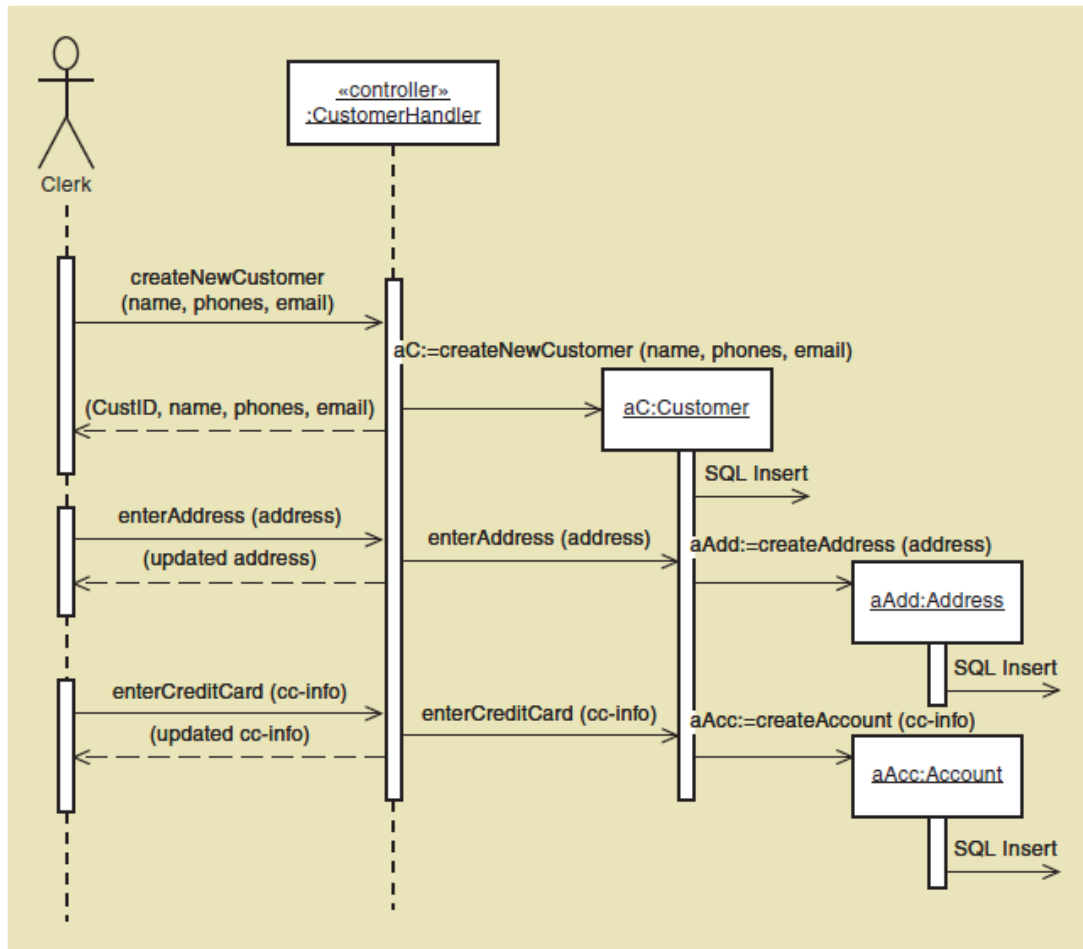
Gambar berikut merupakan contoh system sequence diagram untuk membuat akun pelanggan. Pada diagram dibawah ada tiga pesan input dan dua tanggapan balasan.



**Gambar 2.8 Contoh System Sequence Diagram (Satzinger, Jackson, dan Burd, 2012)**

### 2.1 First Cut System Sequence Diagram

*First Cut Sequence Diagram* digunakan untuk mengidentifikasi semua *class domain* dan diperlukan pesan *internal* antara *actor* dengan *class domain* kemudian ada penambahan pada *view layer* dan *data access layer* (Satzinger, Jackson, & Burd, 2012). Ketika mengidentifikasi dan menciptakan suatu pesan, pertama harus menentukan asal dan tujuan objek tersebut dimana setiap objek akan diperlukan untuk memulai pesan. Setiap pesan harus mencerminkan *request* yang dikirim. Berikut adalah contoh *first cut sequence diagram* yang terdapat pada gambar 2.9.



**Gambar 2.9** Contoh *First Cut System Sequence Diagram* (Satzinger, Jackson, dan Burd, 2012)

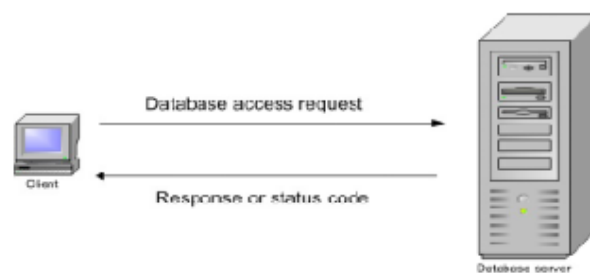
### 2.6.5 Software Architecture

*Software Architecture* terbagi menjadi 2 jenis yaitu *client* dan *server*. *Server* yang berarti sebuah proses, modul, objek atau komputer yang menyediakan layanan melalui jaringan. Sedangkan *client* adalah sebuah proses, modul, objek atau komputer yang meminta layanan dari satu atau lebih *server*. Contohnya: *Three Tier*, karena pada saat *data update* semua bagian dapat melihat perubahan tersebut karena sistem antara bagian-bagian yang mengakses terkomputerisasi sehingga dapat dilihat oleh bagian-bagian yang mengaksesnya. *Software Architecture* memiliki 2 bagian (Satzinger, Jackson dan Burd, 2010) yaitu:

1. *Client/ Server Architecture*, merupakan model umum perangkat lunak organisasi dan dapat diimplementasikan dengan berbagai cara. Ada 2 jenis yaitu:

- a. *Server*: proses, modul, objek, atau komputer yang menyediakan layanan melalui jaringan.
- b. *Client*: Modul, proses, objek, atau komputer yang permintaan jasa dari satu atau lebih *server*.

Berikut adalah contoh *Client/ Server Architecture* yang terdapat pada gambar 2.10.

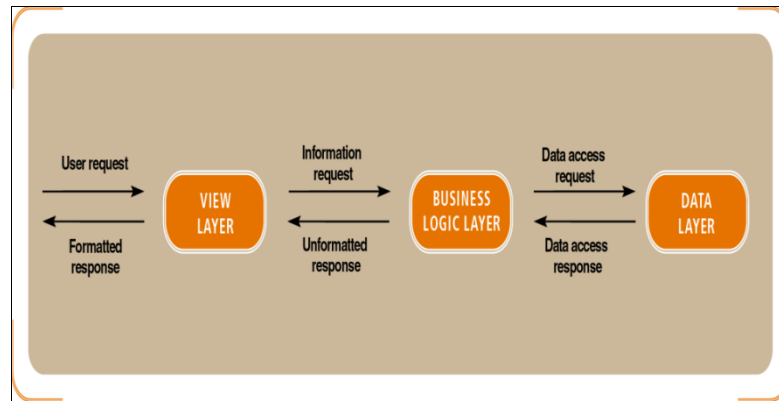


**Gambar 2.10 Contoh *Client/ Server Architecture* (Satzinger, Jackson, dan Burd, 2012)**

2. *Three layer architecture/ server architecture*, merupakan arsitektur *client/server* yang membagi aplikasi ke dalam *data layer*, *business logic layer*, dan *view layer*.

- a. *Data Layer*: bagian dari tiga lapis arsitektur yang berinteraksi dengan *database*.
- b. *Business Logic*: bagian dari tiga arsitektur *layer* yang berisi program-program yang mengimplementasikan aturan bisnis aplikasi.
- c. *View Layer*: bagian dari tiga arsitektur *layer* yang berisi *user interface*.

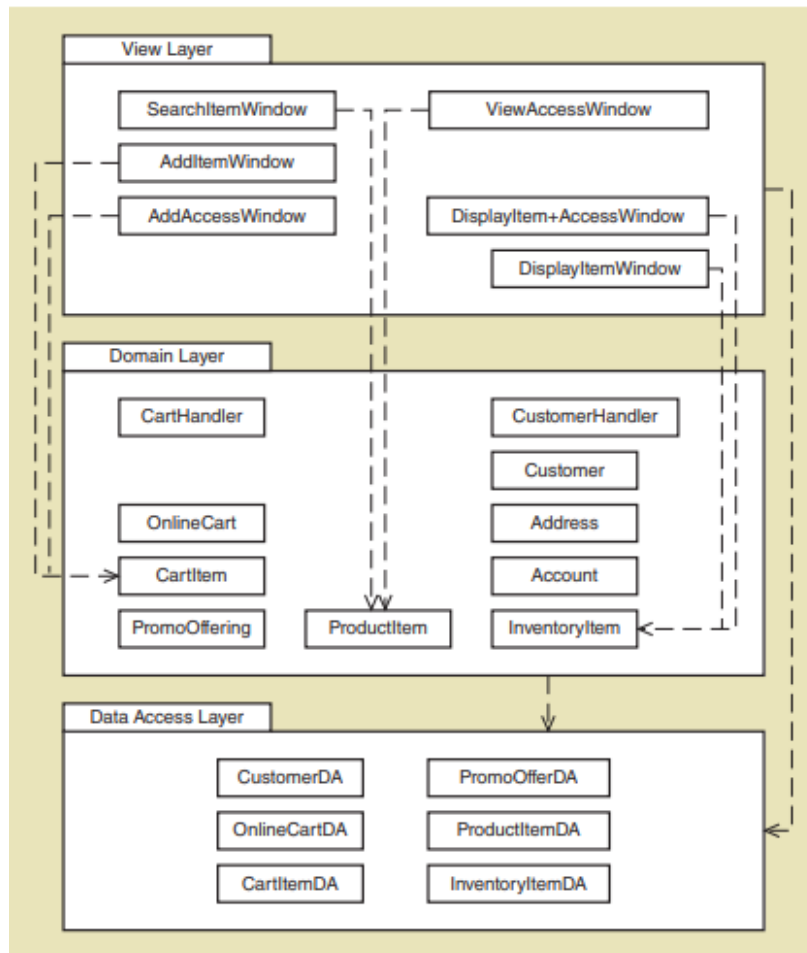
Berikut adalah contoh *Three layer architecture* yang terdapat pada gambar 2.11.



**Gambar 2.11** Contoh *Three Layer Architecture* (Satzinger, Jackson, dan Burd, 2012)

### 2.6.6 Packaged Diagram

*Package diagram* adalah suatu diagram tingkat tinggi yang sederhana yang memungkinkan perancang untuk menghubungkan kelas-kelas dengan grup yang terelasi (Satzinger Jackson dan Burd, 2012). Diagram ini mengilustrasikan *three-design layer*, yaitu *view layer*, *domain layer*, dan *data access layer* dan memperlihatkan setiap lapisan sebagai paket yang terpisah. Berikut adalah contoh *package diagram* yang terdapat pada gambar 2.12.



Gambar 2.12 Contoh *Package Diagram* (Satzinger, Jackson, dan Burd, 2012)

### 2.6.7 Persistent Object

*Persistent object* adalah obyek yang tersedia untuk dapat digunakan sepanjang waktu (Satzinger, Jackson, & Burd, 2012). *Persistent object* disebut juga *relational database* yang digunakan dalam bentuk tabel yang diisi atribut beserta dengan masing – masing nilai dari atribut tersebut. Di setiap tabel tersebut memiliki *primary key*, dimana *primary key* tersebut merupakan sebuah atribut yang unik. Berikut ini adalah contoh *persistent object* yang terdapat pada gambar 2.13.





CatalogID	ProductID	Price	SpecialPrice
23	1244	\$15.00	\$12.00
23	1245	\$15.00	\$12.00
23	1246	\$15.00	\$13.00
23	1247	\$15.00	\$13.00
23	1248	\$14.00	\$11.20
23	1249	\$14.00	\$11.20
23	1252	\$21.00	\$16.80
23	1253	\$21.00	\$16.40
23	1254	\$24.00	\$19.20
23	1257	\$19.00	\$15.20

**Gambar 2.13** Contoh Persistent Object (Satzinger, Jackson, dan Burd, 2012)

### 2.6.8 System Interface

*System interface* merupakan *input* dan *output* yang memerlukan campur tangan manusia (Satzinger, Jackson, & Burd, 2012). *System interface* memungkinkan sebuah inputan ditangkap secara otomatis oleh perangkat input khusus seperti sistem *scanner*, sistem pesan elektronik ke atau dari sistem lain, atau transaksi juga bisa ditangkap oleh sistem lain.

### 2.6.9 User Interface

*User Interface* menggambarkan konversi analisis perancangan sistem menjadi sebuah tampilan yang terdiri dari input dan output antara sistem dan pengguna (Satzinger, Jackson, & Burd, 2012). UI dapat digunakan pengguna internal maupun eksternal dan desainnya bervariasi tergantung dengan kebutuhan yang digunakan. Berikut ini adalah contoh *user interface* yang terdapat pada gambar 2.14.



**Gambar 2.14** Contoh *User Interface* (Satzinger, Jackson, dan Burd, 2012)

### 2.6.10 Personal Home Page (PHP)

PHP merupakan bahasa berbentuk *script* yang ditempatkan dalam *server* dan diproses di *server*. Hasilnya dikirim ke *client*, tempat pemakai menggunakan *browser*. (Kadir, 2002).

PHP adalah *server-side scripting language* yang didesain secara spesifik untuk *web* (Luke Welling and Laura Thomson, 2005). Dalam halaman HTML, dapat dimasukan kode PHP yang akan dieksekusi setiap kali halaman dikunjungi. PHP kemudian diterjemahkan di *web-server* dan diubah menjadi HTML atau *output* lain yang akan dilihat oleh pengunjung halaman. Berikut adalah contoh PHP yg terdapat pada gambar 2.15.

```

<HTML><BODY>
MEMILIH DATABASE<BR>
<?php
    $pemakai="root";
    $password="";
    $id_mysql=mysql_connect("localhost",$pemakai,$password);
    if(!$id_mysql)
        die("DATA BASE GAK BISA DIBUKA");
    if(!mysql_selectdb("coba",$id_mysql))
        die("DATA BASE TAK TERPILIH");
    mysql_close($id_mysql);
    print("SUKSES KONEKSI DAN SELEK DATABASE");
?>
</BODY></HTML>

```

**Gambar 2.15 Contoh PHP (Anisya, 2013)**

### 2.6.10.1 Kelebihan PHP

Terdapat beberapa keuntungan ketika merancang sebuah aplikasi berbasis web dengan menggunakan PHP (Iman, 2013), yaitu :

1. Menjadikan web lebih dinamis.
2. Dapat digunakan oleh siapa saja, karena bersifat *open source*
3. Dapat di jalankan oleh semua jenis sistem operasi.
4. Mendukung banyak paket *database* seperti MySQL.
5. Tidak memerlukan *compile* dalam penggunaannya.
6. Banyak web *server* yang mendukung PHP, seperti Apache dan lain-lain.
7. Pengembangan aplikasi PHP mudah karena banyaknya referensi, dokumentasi dan *developer* yang membantu dalam perkembangannya.

### 1.6.11 Database

Database adalah kumpulan dari data yang tersimpan secara terintegrasi, di-manage dan dikontrol secara terpusat (Satzinger, Jackson dan Burd, 2012).

Database terdiri dari dua bagian yang berhubungan; yaitu physical data store dan schema (Satzinger, Jackson dan Burd, 2012).

Physical data store menampung data bit dan byte mentah yang dibuat dan digunakan oleh sistem informasi (misalnya nama, harga, saldo). Sedangkan schema menampung informasi deskriptif tentang data yang ditampung dalam physical data store, yang mencakup:

- Organisasi akan data individual yang tersimpan dalam tabel.
- Asosiasi antara tabel atau class.
- Detail akan organisasi physical data store, mencakup type, lengths, location, dan indexing dari data tersebut.
- Pengendalian akan akses dan konten, mencakup nilai data yang diperbolehkan untuk suatu data, ketergantungan nilai antar data, dan daftar pengguna yang diperbolehkan untuk melakukan read atau write terhadap suatu data tertentu.

#### **2.6.11.1 MySQL**

*MySQL* mungkin adalah sistem manajemen basis data yang paling populer untuk *server web* (Nixon, 2015). *MySQL* juga sangat terukur. Basis data adalah kumpulan catatan atau data terstruktur yang tersimpan dalam sistem komputer dan diatur sedemikian rupa sehingga mudah dicari dan informasi dapat dengan cepat diambil. *SQL* di *MySQL* adalah singkatan dari *Structured Query Language*. Bahasa ini berbasis bahasa Inggris yang juga digunakan di basis data lain seperti Oracle dan *Microsoft SQL Server*. Basis data *MySQL* berisi satu atau lebih tabel, yang masing-masing berisi catatan atau baris. Dalam baris ini ada berbagai kolom atau *field* yang berisi data itu sendiri. Berikut adalah contoh basis data yang terdapat pada gambar 2.16.

Author	Title	Type	Year
Mark Twain	The Adventures of Tom Sawyer	Fiction	1876
Jane Austen	Pride and Prejudice	Fiction	1811
Charles Darwin	The Origin of Species	Non-Fiction	1856
Charles Dickens	The Old Curiosity Shop	Fiction	1841
William Shakespeare	Romeo and Juliet	Play	1594

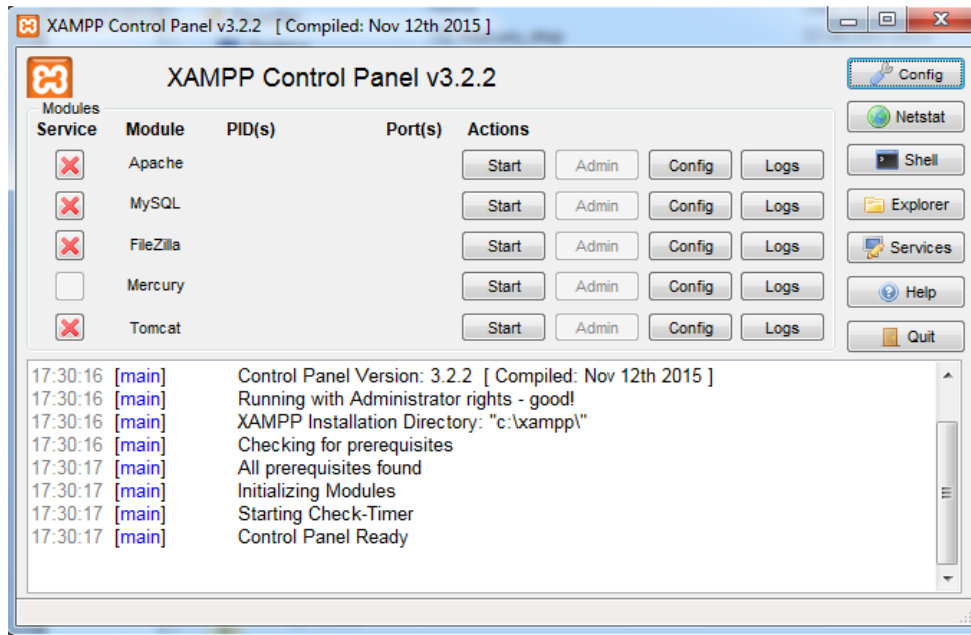
**Gambar 2.16 Contoh Basis Data (Nixon, 2015)**

#### 2.6.11.2 XAMPP

XAMPP adalah sebuah *software* yang berfungsi untuk menjalankan *website* berbasis *PHP* dan menggunakan pengolah data *MySQL* dikomputer lokal (Yogi Wicaksono, 2008). XAMPP berperan sebagai *server web* pada komputer.

XAMPP adalah suatu paket yang dibentuk oleh *Apache Friend*, yang terdapat beberapa *software* juga yang berupa *Apache*, *PHP*, *MySQL*, dan banyak yang lainnya (Sirowich, Darie, 2007).

Berikut adalah contoh *xampp* yang terdapat pada gambar 2.17.



**Gambar 2.17 XAMPP**