BAB 2
LANDASAN TEORI

2.1 Sistem Informasi

Sistem adalah sekumpulan prosedur yang saling terintegrasi yang memiliki maksud yang sama untuk menyelesaikan suatu tujuan (Satzinger et al, 2012). Sistem adalah sekelompok komponen yang saling berhubungan, dengan batasan yang jelas, dan bekerja sama menuju tujuan tertentu dengan menerima *input* serta menghasilkan *output* yang merupakan fungsi dasar dalam proses transformasi yang teratur (O'Brien, 2013).

Berdasarkan pengertian diatas dapat disimpulkan bahwa sistem merupakan kumpulan komponen, prosedur yang terintegrasi dan bekerja bersama sama untuk mencapai tujuan tertentu.

Informasi didefinisikan sebagai data yang disajikan dalam bentuk yang membantu dalam aktivitas pengambilan keputusan. Informasi tersebut mempunyai nilai kepada pengambil keputusan karena mengurangi ketidakpastian dan meningkatkan pengetahuan akan area tertentu yang menjadi perhatian (Gelinas dan Dull, 2012).

Sistem informasi merupakan sekumpulan komponen komputer yang saling terkait untuk mengumpulkan, memproses, menyimpan dan menyediakan *output* informasi yang dibutuhkan untuk menyelesaikan pekerjaan bisnis (Satzinger et al, 2012).

Dari beberapa definisi diatas dapat disimpulkan bahwa sistem informasi merupakan kumpulan komponen yang terintegrasi untuk mengumpulkan, memproses, menyimpan dan menyediakan *output* informasi yang berguna bagi organisasi atau perusahaan.

2.1.1 Komponen Sistem Informasi

Sistem informasi secara luar terdiri dari berbagai komponen serta orang-orang yang bertugas untuk mengelola dan mendistribusikan informasi. Setiap komponen sistem informasi saling bekerja sama dan terhubung satu sama lain sehingga sistem informasi bisa berjalan

dengan baik. Adapun komponen-komponen sistem informasi terdiri dari 5 (lima) sebagai berikut (O'Brien et al, 2011):

1. Sumber daya manusia (*Human Resource*)

   Sumber daya manusia merupakan komponen penting dan dibutuhkan dalam pengelolaan sistem informasi, sumber daya manusia meliputi pengguna akhir dan pengguna spesialis.

2. Sumber daya perangkat keras (*Hardware*)

   Perangkat keras merupakan perangkat fisik dan bahan yang digunakan dalam mengelola sistem informasi.

3. Sumber daya perangkat lunak (*Software*)

   Perangkat lunak mencakup semua perintah dalam pemrosesan informasi, perangkat lunak meliputi rangkaian perintah dengan *hardware* komputer yang disebut program.

4. Sumber daya data (*Data resource*)

   Dalam sistem informasi data diolah dan menghasilkan informasi, data bisa ditampilkan dalam banyak bentuk misalnya angkat, huruf, teks, gambar, grafik, suara, video maupun karakter lain yang menggambarkan transaksi bisnis, kejadian dan entitas lain.

5. Sumber daya jaringan (*Network resource*)

   Sumber daya jaringan sangat penting dalam sistem informasi, jaringan menghubungkan peralatan antara satu dengan yang lain dalam pemrosesan informasi dan dikendalikan melalui perangkat lunak.

2.1.2 Karakteristik Sistem Informasi

Sebuah sistem dapat kita katakan sebagai sistem informasi apabila memenuhi karakteristik utama dari sebuah sistem informasi. Karakteristik utama tersebut menunjukkan bahwa sebuah sistem harus mampu memberikan arus informasi dari *host* menuju pengguna. Adapun beberapa karakteristik yang dimiliki oleh sistem informasi sebagai berikut:

a. Memiliki komponen, komponen ini merupakan bagian dari sebuah sistem informasi, dimana keseluruhan komponen tersebut saling berinteraksi satu sama lain. Setiap komponen

memiliki sifat untuk menjalankan fungsi-fungsi tertentu di dalam sebuah sistem informasi.

b. Memiliki batasan, batasan merupakan pembatas dari sebuah sistem informasi dengan sistem informasi lainnya, yang menunjukkan ruang lingkup yang dimiliki.
c. Memiliki lingkungan luar (*Environment*), *environment* merupakan keseluruhan sistem dan juga lingkungan yang berada di luar batasan atau boundary dari sebuah sistem informasi.
d. Memiliki *interface, interface* atau antarmuka merupakan media yang digunakan untuk menghubungkan sebuah komponen yang terdapat pada sebuah sistem informasi.
e. Memiliki masukan (*input*), *input* merupakan data yang dimasukkan ke dalam suatu sistem untuk diolah.
f. Memiliki keluaran (*output*), *output* merupakan keluaran dari data yang sudah diolah berupa informasi. Informasi yang dikeluarkan berguna bagi pengguna.
g. Memiliki pemrosesan, pemrosesan merupakan komponen di dalam sebuah sistem informasi yang bertugas untuk memproses masukan menjadi keluaran.
h. Memiliki sasaran, sasaran merupakan hal yang harus ditentukan dan dicapai dengan menggunakan sistem informasi, sebuah informasi dianggap berhasil apabila dapat mencapai tujuan yang sudah ditentukan.

2.2 Gift

Gift dalam bahasa indonesia berarti “hadiah”. Kamus Besar Bahasa Indonesia mengartikan hadiah sebagai pemberian, persenan, atau kado. Hadiah merupakan sarana untuk menunjukkan ekspresi emosional dan kepribadian atau diri pemberi hadiah (Wolfinbarger, 1990). Hadiah menginterpretasikan sebagai sebuah ajakan pertemanan dan keterlibatan pemberi dalam peristiwa suka dan duka yang dialami penerima (Jr. John, 1983). Hadiah juga dapat dijadikan sebagai dasar penciptaan dan pembangunan kembali relasi antara pemberi dan penerima, misalnya ucapan

terima kasih, permintaan maaf, simbolisasi cinta, dan berbagai ekspresi emosional yang lain.

Pemberian hadiah merupakan aktivitas manusia yang telah mendunia. Praktik kuno pemberian hadiah secara signifikan mempengaruhi budaya di seluruh dunia dan berkembang menjadi sebuah proses menciptakan dan mengelola hubungan sosial (Sherry, 1983). Aktivitas memberi hadiah dalam peristiwa tertentu, seperti perayaan ulang tahun, pernikahan, Natal, Valentine, dan sebagainya, telah biasa dilakukan oleh berbagai orang di seluruh dunia. Mauss mengatakan, pentingnya hadiah biasanya tidak hanya untuk mentransfer objek hadiah itu sendiri, tetapi juga aspek sosial, yaitu keharusan untuk memberi, menerima, dan saling membalas pemberian hadiah.

2.3 *Funding*

Funding atau disebut penghimpunan dana adalah kegiatan mengumpulkan dana dari masyarakat yang dilakukan oleh organisasi dalam bentuk tabungan atau simpanan (Pandia, 2013). Dalam penghimpunan dana ini dilakukan dengan perencanaan yang matang dan waktu yang telah ditentukan. Penghimpunan dana memberikan manfaat bagi berbagai pihak, seperti: bank, pengelola dana, dan penerima dana.

2.4 *Gift Funding*

Gift Funding merupakan terminologi dari "*gift"* (hadiah) dan *Funding* (penghimpun dana). Dengan demikian, arti keseluruhan *gift funding* adalah penghimpunan dana yang diberikan oleh individu berdasarkan jumlah yang sesuai dengan hadiah yang akan diberikan. Pemberian dana yang diberikan bervariasi, semakin besar dana yang diberikan maka semakin banyak dana yang terkumpul untuk tercapainya hadiah yang akan diberikan. Contoh kitabisa.com berbasis donasi dengan berbagai kategori.

2.5 *System Development Life Cycle* (SDLC)

System Development Life Cycle (SDLC) adalah suatu pengembangan sistem yang mengidentifikasi semua kegiatan yang diperlukan untuk membangun, meluncurkan dan memelihara sistem informasi (Satzinger et al, 2012). *System Development Life Cycle* (SDLC) adalah pendekatan yang dilakukan secara bertahap dalam hal melakukan analisa dan membangun

rancangan sistem dengan menggunakan siklus-siklus secara spesifik terhadap kegiatan penggunanya.

Sehingga berdasarkan definisi diatas, dapat dijelaskan bahwa *System Development Life Cycle* merupakan metode yang dilakukan dalam membangun sistem informasi melalui beberapa fase bertahap mulai dari perencanaan sampai dengan implementasi. Pada penelitian ini metode analisis yang digunakan yaitu dengan menggunakan model *Waterfall*.

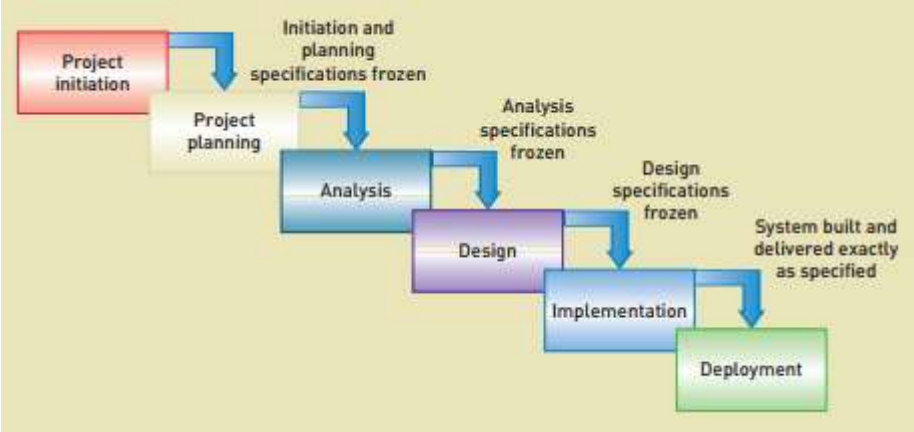
2.6 *Object-Oriented Analysis and Design* (OOAD)

Object Oriented Analysis (OOA) mendefinisikan semua jenis objek yang melakukan pekerjaan dalam sistem dan menunjukkan interaksi pengguna yang disebut juga sebagai case yang diperlukan untuk menyelesaikan tugas (Satzinger et al, 2012).

Object Oriented Design (OOD) mendefinisikan semua jenis tambahan objek yang diperlukan untuk berkomunikasi dengan orang-orang dan perangkat dalam sistem, menunjukkan bagaimana objek berinteraksi untuk menyelesaikan tugas- tugas, dan memurnikan definisi dari setiap jenis objek sehingga dapat diimplementasikan dengan bahasa atau lingkungan tertentu (Satzinger et al, 2012).

2.7 Model *Waterfall*

Model *Waterfall* menggambarkan pendekatan sekuensial beberapa tahap yang biasanya disebut juga dengan model air terjun. Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau urut dimulai dari analisis, desain, pengkodean, pengujian dan tahap *support* (Satzinger et al, 2012).



Gambar 2.1 Model *Waterfall* (Satzinger et al, 2012)

Berdasarkan gambar tersebut terdapat 6 (enam) proses inti yang diperlukan dalam pengembangan sistem yaitu sebagai berikut:

1. Identifikasi permasalahan atau kebutuhan dan mendapatkan persetujuan untuk memulai *project*.
2. Merencanakan dan memantau jalannya proyek. Hal-hal yang perlu direncanakan terkait apa yang harus dilakukan (*what-to-do*), bagaimana melakukannya (*how-to-do it*), dan siapa yang melakukannya (*who-does-it*).
3. Menemukan dan memahami detil-detil dari permasalahan dan kebutuhan.
4. Merancang komponen-komponen sistem guna menyelesaikan permasalahan atau memenuhi kebutuhan.
5. Membangun, melakukan uji coba, dan mengintegrasikan komponen-komponen sistem.
6. Menyelesaikan kegiatan uji coba sistem dan mengimplementasikan sistem tersebut.

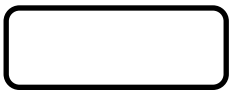
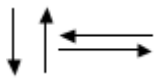
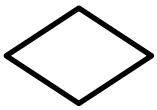
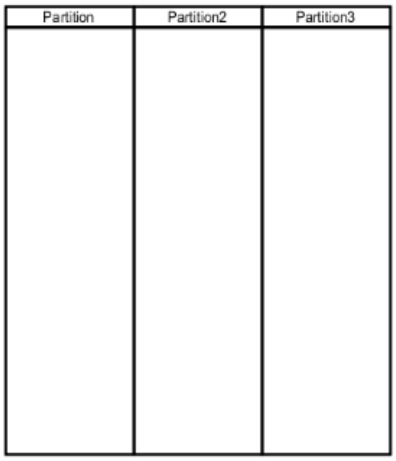
2.8 *Unified Modeling Language* (UML)

Unified modelling language (UML) merupakan kumpulan standar model konstruksi dan notasi yang didefinisikan oleh *object management group (OMG)*, sebuah organisasi standar untuk pengembangan sistem (Satzinger et al, 2012). Dengan menggunakan UML, analisis dan pengguna akhir dapat menggambarkan dan memahami berbagai diagram khusus yang digunakan dalam pengembangan sistem.

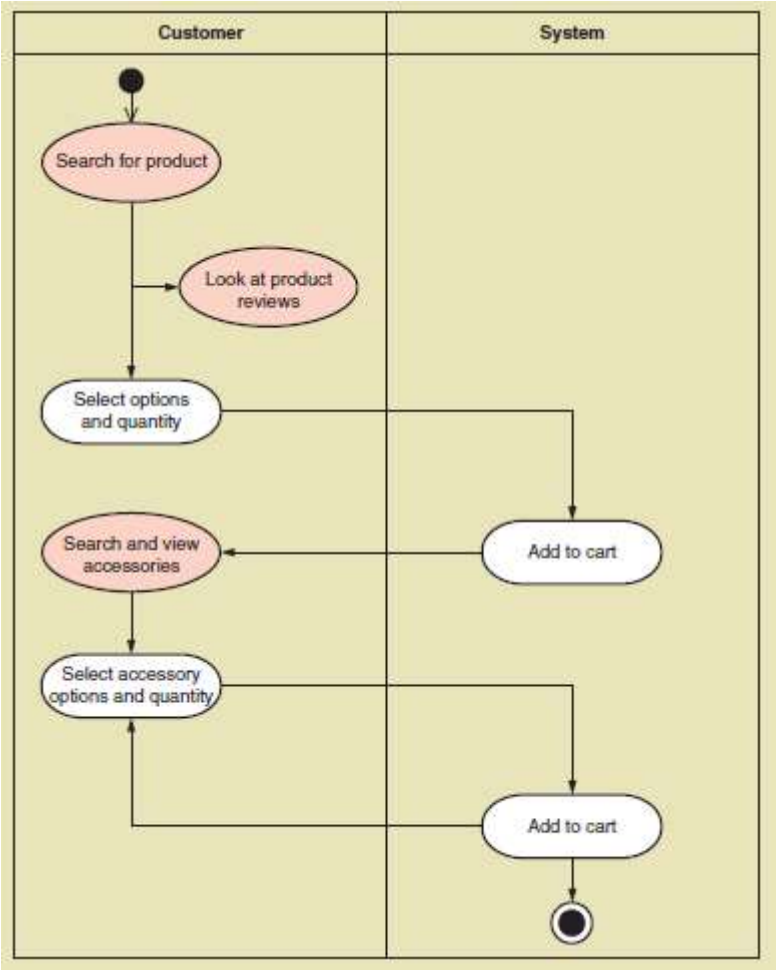
2.8.1 Activity Diagram

Activity diagram merupakan diagram alur kerja yang menggambarkan berbagai kegiatan pengguna (atau sistem), orang yang melakukan setiap kegiatan, dan aliran sekuensial kegiatan tersebut. *Activity diagram* adalah salah satu diagram *Unified Modeling Language* (UML) yang terkait dengan pendekatan berorientasi objek, tetapi dapat digunakan dengan pendekatan lain (Satzinger et al, 2010).

Tabel 2.1 Simbol *Activity Diagram* (Satzinger et al, 2010)

Simbol	Nama dan Keterangan
	**Activity** menggambarkan sebuah pekerjaan atau tugas dalam workflow
	**Start date** menunjukan dimulai workflow pada suatu activity diagram
	**End state** menunjukan akhir atau terminal pada activity diagram
	**State transition** menunjukan kegiatan selanjutnya setelah kegiatan yang dilakukan
	**Decision** suatu titik pada activity diagram yang menunjukan kondisi adanya perbedaan transisi
	**Object swimlaness** menunjukan objek yang bertangung jawab pada aktivitas tertentu

Berikut ini contoh *activity diagram* ketika *customer* menambahkan barang atau berbelanja pada toko *online*. Prosesnya dimulai ketika *customer* mencari barang kemudian me-*reviews* hasil pencarian dan menambahkan jumlah barang yang akan dibeli pada keranjang (*cart*).

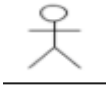
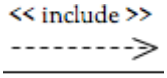
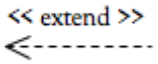


Gambar 2.2 Contoh *Activity Diagram* (Satzinger et al, 2012)

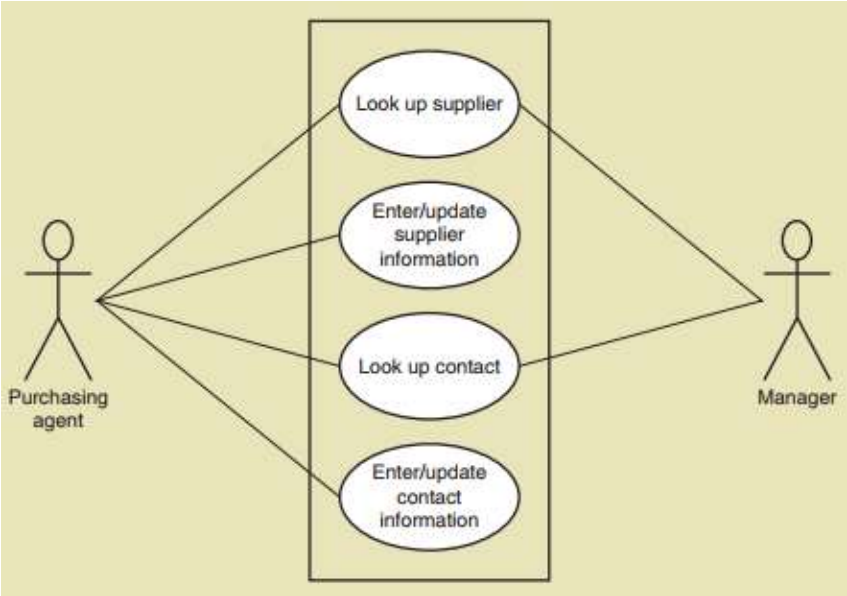
2.8.2 *Use Case Diagram*

Use case diagram merupakan sebuah diagram yang digunakan untuk menggambarkan interaksi antara pengguna dengan sistem. Pada *use case diagram* menekankan pada "siapa" melakukan "apa" dalam lingkup sistem yang sedang dibangun. Komponen-komponen seperti *actor*, *use-case* saling berinteraksi sehingga memberikan gambaran bagaimana sistem tersebut bekerja. Berikut beberapa simbol dalam *use case diagram* menurut (Satzinger et al, 2010):

Tabel 2.2 Simbol *Use Case Diagram* (Satzinger et al, 2010)

Simbol	Nama	Keterangan
	*Actor*	Merupakan pengguna atau orang yang berinteraksi dengan *use case*
	*Include*	Menspesifikasikan bahwa *use case* sumber secara eksplisit
	*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan
	Association	Apa yang menghubungkan antara objek satu dengan objek yang lain
	*Sistem*	Menspesifikasikan paket yang menampilkan sistem secara terbatas
	*Use Case*	Mendeskripsikan urutan aksi-aksi yang ditampilkan sistem dan menghasilkan suatu hasil yang terukur bagi actor

Berikut contoh *use case diagram* yang terjadi pada proses *purchasing* yang melibatkan dua *actor* yaitu, *purchasing agent* dan *manager*, *user case diagram* menggambarkan aktivitas yang bisa dilakukan masing-masing *actor* dalam sistem.



Gambar 2.3 Contoh *Use Case Diagram* (Satzinger et al, 2012)

2.8.3 *Use Case Description*

Use case adalah 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. Informasi rinci tentang setiap kasus penggunaan digambarkan dengan menggunakan deskripsi kasus (Satzinger et al, 2012)

Dalam penelitian ini terdapat dua jenis *use case description* yang akan digunakan, berikut adalah penjelasan singkatnya.

1. *Brief Use Case Description* dapat digunakan untuk *use case* yang sederhana, khususnya sistem yang dikembangkan untuk aplikasi kecil yang mudah dimengerti.

Use case	Brief use case description
Create customer account	User/actor enters new customer account data, and the system assigns account number, creates a customer record, and creates an account record.
Look up customer	User/actor enters customer account number, and the system retrieves and displays customer and account data.
Process account adjustment	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 2.4 Contoh *Brief Use Case Description* (Satzinger et al, 2012)

2. *Fully Developed Use Case Description* merupakan metode untuk mendokumentasikan *use case* karena menjelaskan secara rinci

setiap proses dalam *use case diagram*. Adapun detail dari *Fully Developed Use Case Description* yang menjelaskan beberapa hal sebagai berikut:

a. *Use case name*, harus diisi dengan nama dari *use case* yang akan dijelaskan di *use case description*.
b. *Scenario*, merupakan nama skenario dari *use case* yang akan dijelaskan. Pada umumnya nama skenario dapat disamakan dengan nama dari *use case*.
c. *Triggering event*, merupakan event yang memicu terjadinya *use case*. Anda dapat melihat referensi untuk event ini dari event table.
d. *Brief description*, merupakan ringkasan singkat secara umum dari *use case* yang akan dijelaskan.
e. *Actors*, merupakan para pelaku atau pengguna sistem yang terkait dengan jalannya sebuah *use case*.
f. *Related use cases*, merupakan *use cases* lain yang terkait atau akan dijalankan apabila *use cases*, yang akan dibahas, dieksekusi.
g. *Stakeholders*, merupakan pihak-pihak yang berkepentingan terkait dengan jalannya *use case* yang akan dijelaskan.
h. *Preconditions*, merupakan kondisi awal yang harus terpenuhi agar eksekusi dari *use cases*, yang akan dijelaskan dapat terlaksana.
i. *Postconditions*, merupakan kondisi yang terjadi setelah *use cases* selesai dieksekusi.
j. *Flow of activities*, mendeskripsikan rincian aliran aktivitas-aktivitas yang terlaksana dari *use cases* yang akan dijelaskan. Selain aktivitas-aktivitas tersebut akan digambarkan secara berurutan, *flow of activities* juga akan menggambarkan interaksi aktivitas antara yang dilakukan oleh *actors* dengan *system*.
k. *Exception condition*, mendeskripsikan aktivitas-aktivitas khusus yang terjadi apabila suatu kondisi terpenuhi pada saat eksekusi dari sebuah use case.

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

Gambar 2.5 Contoh *Fully Developed Use Case Description* (Satzinger et al, 2012)

2.8.4 *Domain Model Class Diagram*

Class diagram digunakan untuk menunjukkan objek *class* untuk sistem. Notasinya dari *Unified Modeling Language* (UML), yang telah menjadi standar untuk model yang digunakan dengan pengembangan *system object oriented* (Satzinger et al, 2010).

Salah satu jenis class diagram UML menunjukkan hal-hal dalam pekerjaan *domain user* disebut sebagai domain *model class diagram*. Di class diagram, persegi panjang mewakili kelas, dan garis yang menghubungkan persegi panjang menunjukkan asosiasi antara kelas. Dalam persegi panjang (kotak) terbagi dua, bagian atas berisi nama kelas, dan bagian bawah merupakan atribut kelas. Nama kelas selalu diawali dengan huruf kapital, dan atribut nama selalu diawali dengan huruf kecil. *Diagram class* digambarkan dengan menampilkan kelas dan asosiasi antara kelas.

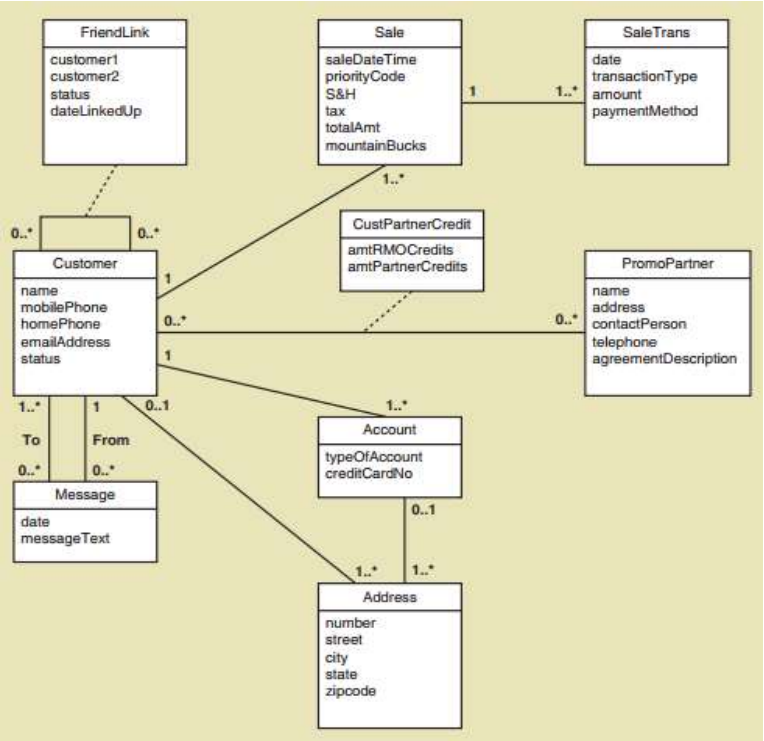
Berdasarkan penjelasan diatas, maka dapat disimpulkan bahwa *Domain Model Class Diagram* digunakan untuk mendefinisikan kelas yang ada pada notasi UML (*Unified Modeling Language*).

Tabel 2.3 Simbol *Domain Class Diagram* (Satzinger et al, 2010).

Simbol	Nama	Keterangan
	Class	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada diatasnya objek induk (*ancestor*)
	Association	Menggambar relasi asosiasi 0..1 zero or one 0..* zero or more 1 one and only 1..1 one and only one 1..* one or more
	Association class	Menghubungkan kelas asosiasi *(association class)* pada suatu relasi asosiasi
	Generalization	Menggambarkan relasi generalisasi
	Realize	Menggambarkan relasi realisasai
	Aggregation	Menggambarkan relasi agregasi
	Composition	Menggambarkan sebuah class tidak bisa berdiri sendiri dan harus jadi bagian dari class yang lain

Berikut contoh *Domain Class Diagram* dalam *subsistem* penjualan, Subsistem *Customer* mencakup *messages*, *partner credits*, dan *friend links*. Kelas *FriendLink* adalah kelas asosiasi, tetapi tidak seperti contoh lainnya, kelas *FriendLink* melekat pada hubungan *unary* antara *Customer*. Setiap *Customer* bisa dihubungkan dengan

banyak *Customer* lain, ditunjukkan oleh garis asosiasi di bagian atas kelas *Customer*. Setiap *Customer* dapat mengirim banyak pesan, masing-masing untuk banyak *Customer* lain. Demikian pula, setiap *Customer* dapat menerima banyak pesan.

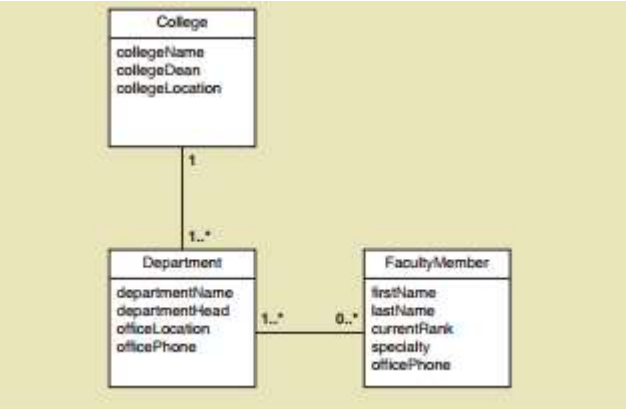


Gambar 2.6 Contoh *Domain Class Diagram* (Satzinger et al, 2010)

2.8.5 *Update Design Class Diagram*

Design class diagram dapat dirancang untuk setiap layer. Tahap pertama di dalam *updated design class diagram* adalah menambahkan method berdasarkan informasi dari sequence diagram. Terdapat tiga jenis *method* yaitu *Contructor Method*, *Data-get* dan *data-set method* dan terakhir adalah *Use-case spesific method*. Untuk menghindari kelebihan informasi, pengembang biasanya tidak memasukkan method get dan set untuk setiap design class diagram (Satzinger et al, 2010).

Berikut contoh *Update Design Class Diagram* untuk universitas, terdapat 3 (tiga) class yaitu *College*, *Department*, *FacultyMember*.

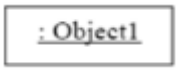
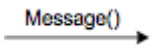


Gambar 2.7 Contoh *Update Design Class Diagram* (Satzinger et al, 2012)

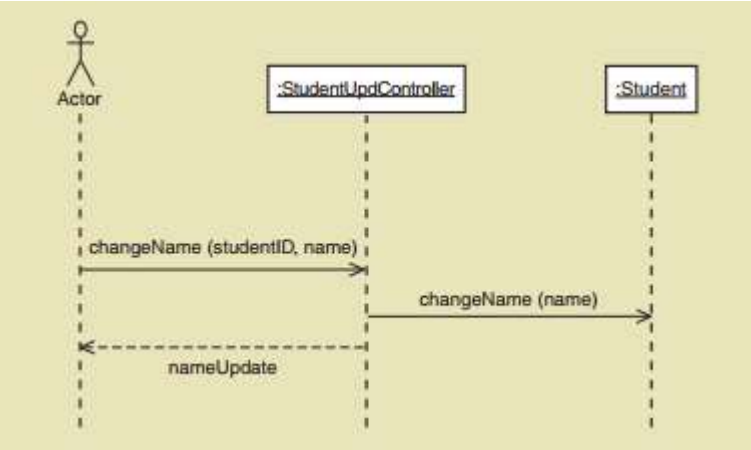
2.8.6 *System Sequence Diagram*

System sequence diagram digunakan untuk menggambarkan aliran dari informasi yang masuk dan keluar dari sistem yang terotomatisasi (Satzinger et al, 2012). Dalam *System sequence diagram* terdapat input dan output yang mengidentifikasi interaksi antara aktor dan sistem.

Tabel 2.4 Simbol *System Sequence Diagram* (Satzinger et al, 2012)

Simbol	Nama	Keterangan
	*Object*	Merupakan *instance* dari sebuah class dan dituliskan secara horizontal, digambarkan sebagai sebuah class dengan nama objek didalamnya
	*Actor*	*Actor* dapat berkomunikasi dengan objek, actor dapat diurutkan sebagai kolom
	Lifeline	Mengindikasi keberadaan sebuah objek dalam basis waktu, notasinya garis putus-putus yang ditarik dari sebuah objek
	*Message*	Komunikasi antara objek yang memuat informasi-informasi tentang aktivitas yang terjadi
	*Activation*	Mengindikasikan sebuah objek yang akan melakukan aksi

Berikut contoh *System Sequence Diagram* dalam proses *updating student name*.

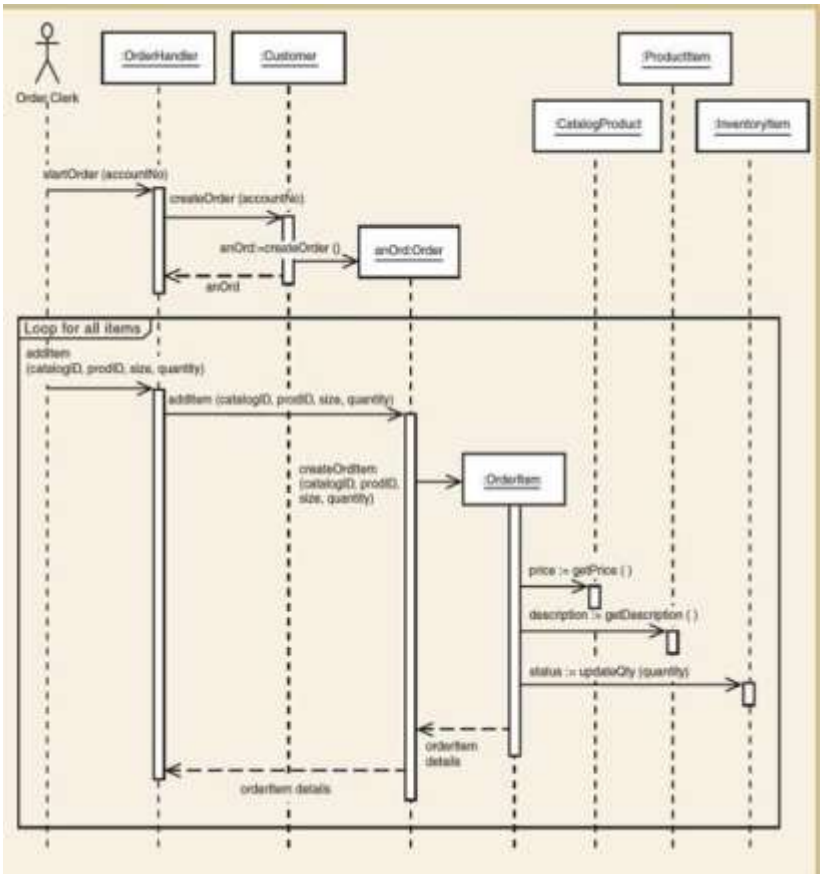


Gambar 2.8 Contoh *System Sequence Diagram* (Satzinger et al, 2012)

System Sequence Diagram merupakan sebuah Diagram yang menunjukkan eksekusi *operation* pada sebuah objek yang melibatkan pemanggilan *operation* di objek lain.

2.8.7 *First Cut 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*. 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 (Satzinger et al, 2012). Berikut contoh *first cut sequence diagram* ketika *customer* melakukan pemesanan pada toko *online*.

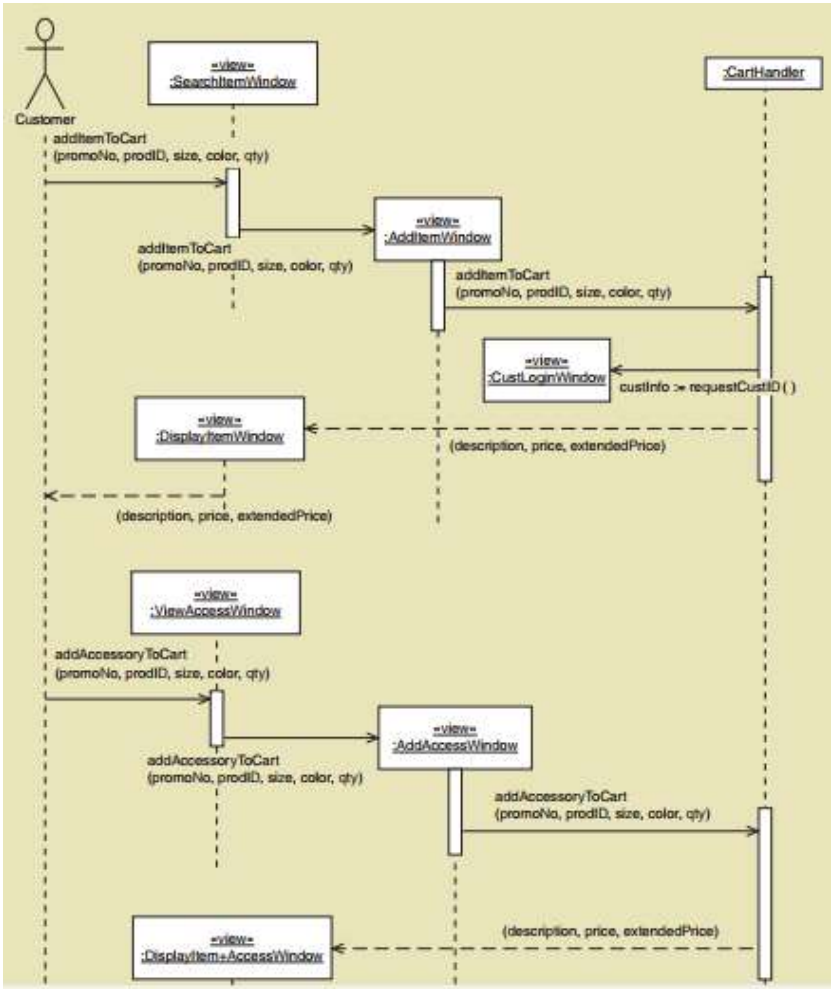


Gambar 2.9 Contoh *First Cut Sequence Diagram* (Satzinger et al, 2012)

2.8.8 *View Layer*

View Layer adalah suatu bagian dari *three-layer architecture* yang memiliki isi yaitu *user interface* dan berfungsi untuk menerima *input user*, memformatnya, lalu menampilkan hasil proses. (Satzinger et al, 2012).

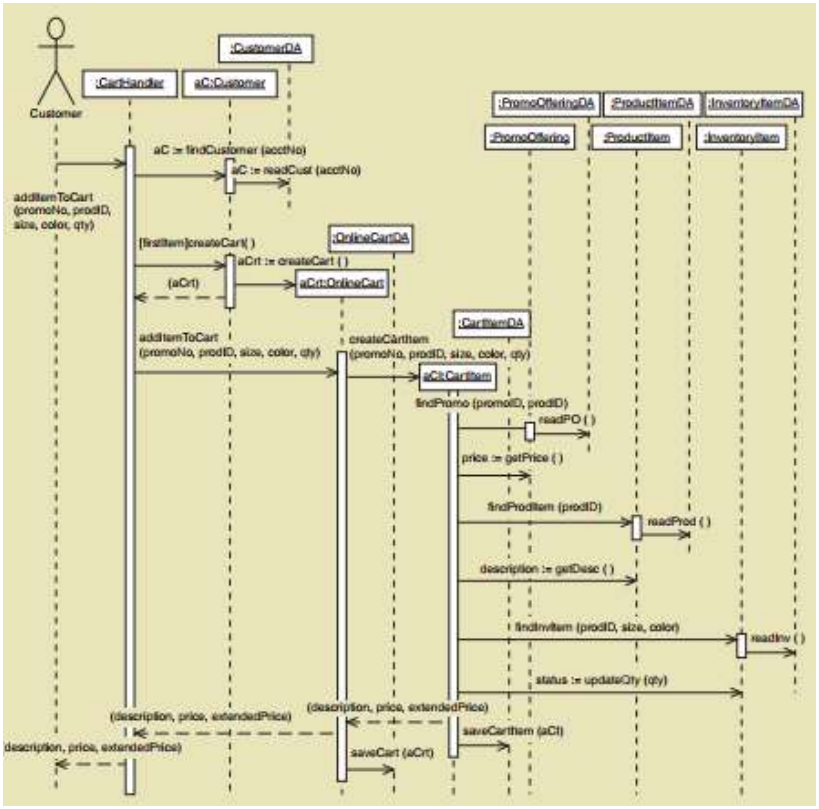
Berikut contoh *view layer* ketika *Customer* melakukan pembelian barang dari sebuah toko *online*.



Gambar 2.10 Contoh *View Layer* (Satzinger et al, 2010)

2.8.9 *Data Access Layer*

Data access layer merupakan bagian dari *three-layer architecture* yang memiliki interaksi dengan *database* serta memiliki tugas untuk mengelola data yang disimpan pada satu atau lebih *database*. Tahapan dalam membuat data *access layer* yaitu dengan menentukan *domain objects* dengan data yang diperlukan dari *database*, kemudian lakukan *query database* dan mengirim *object reference*, dan yang terakhir lakukan proses pengembalian informasi dalam referensi *object*. (Satzinger et al, 2010). Berikut contoh *data access layer* pada proses pemesanan barang yang dilakukan *Customer* pada toko *online*.

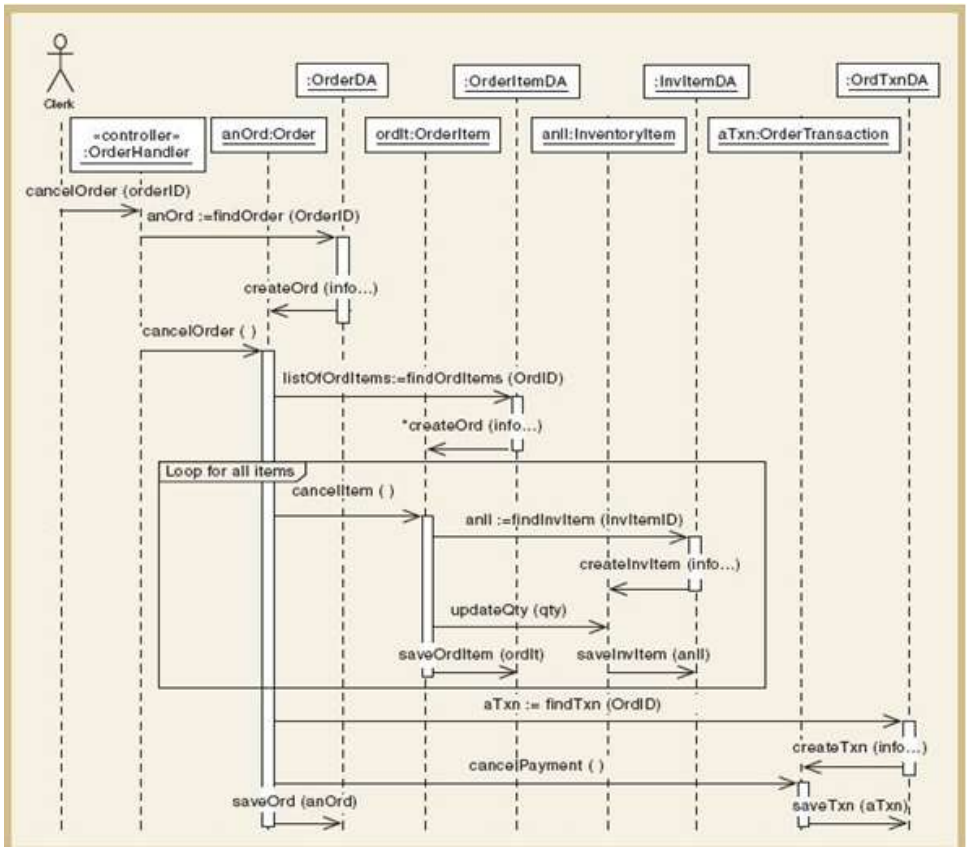


Gambar 2.11 Contoh *Data Access Layer* (Satzinger et al, 2010)

2.8.10 *Multilayer*

Multilayer memiliki tujuan utama untuk mengidentifikasi kolaborasi kelas dan apakah kelas tersebut harus mengirim pesan antara satu sama lain. (Satzinger et al, 2012).

Contoh gambar *multilayer diagram* ditunjukkan pada Gambar 2.13. diagram tersebut menggambarkan sebuah proses pembelian / pemesanan pada *e-commerce*.



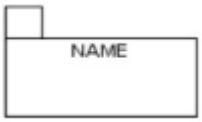
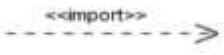
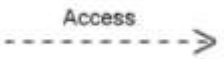
Gambar 2.12 Contoh *Multilayer* (Satzinger et al, 2012)

2.8.11 *Package Diagram*

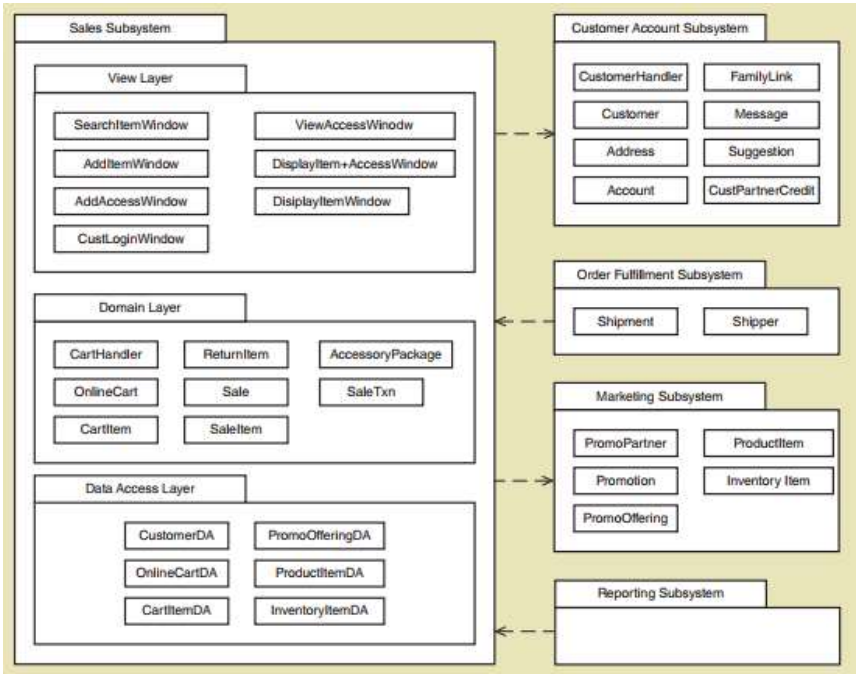
Package diagram adalah suatu diagram tingkat tinggi dalam bentuk sederhana yang memungkinkan perancang untuk menghubungkan kelas-kelas dengan grup yang berelasi (Satzinger et al, 2012).

Package diagram digunakan untuk mendokumentasikan perbedaan atau kesamaan dalam hubungan objek pada tiga layer perancangan, yaitu *view*, *domain* dan *data access*.

Tabel 2.5 Simbol *Package Diagram* (Satzinger et al, 2010)

Simbol	Nama	Keterangan
	*Package*	Merupakan sebuah elemen atau lebih dari kelas
	*Import*	Suatu dependency yang mengindikasikan isi tujuan paket secara umum yang ditambahkan ke dalam sumber paket
	*Access*	Suatu dependency yang mengindikasikan isi tujuan paket secara umum yang digunakan pada nama sumber paket

Berikut contoh *package diagram* pada proses penjualan pada sebuah perusahaan.



Gambar 2.13 Contoh *Package Diagram* (Satzinger et al, 2012)

2.8.12 *Persistent Object*

Persistent Object adalah objek yang tersedia pada sistem untuk diingat dan digunakan dari waktu (Satzinger et al, 2012). Berikut contoh *persistent object* data mahasiswa pada suatu universitas.

Tabel 2.6 Contoh Tabel *Persistent Object* (Satzinger et al, 2012)

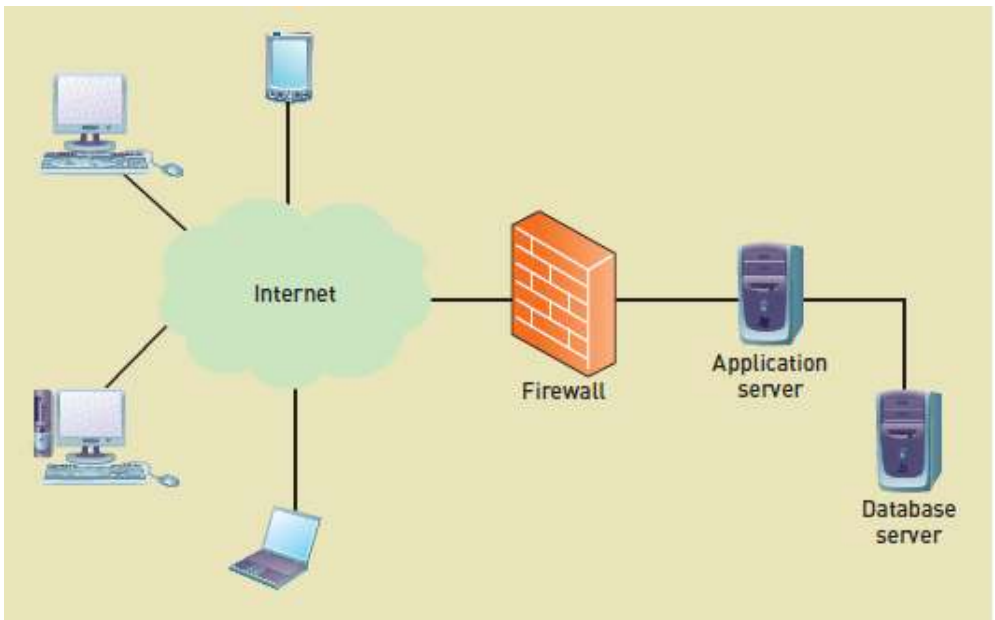
CatalogProduct : Table

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

Record: 11 of 11

2.9 *Network Environment*

Dalam melakukan deployment *environment* terdiri dari *hardware*, *software* dan lingkungan *network* dimana sistem akan dioperasikan. Aktivitas pertama adalah menyiapkan infrastruktur berupa *network*, sistem yang menjalankan perangkat lunak dan kemudian akan terkoneksi langsung dengan internet (Satzinger et al, 2012).



Gambar 2.14 Contoh Diagram *Network Environment* (Satzinger et al, 2012)

2.10 Aplikasi Model (*Web Base*)

2.10.1 *Website*

Website merupakan layanan perangkat lunak ke dalam proses server yang dapat diakses melalui protokol web, termasuk *XML*, *SOAP*, *Web Services Description Language* (WSDL), dan *Universal Description, Discovery,* and *Integration* (UDDI).

2.10.2 *Personal Home Page* (PHP)

PHP adalah suatu bahasa pemrograman yang difungsikan untuk membangun suatu website**.** PHP atau singkatan dari *Personal Home Page* merupakan bahasa *script* yang tertanam dalam *HTML* untuk dieksekusi bersifat *server-side* (Nugroho, 2006). Karena tergolong dalam bahasa pemrograman *open-source*, sumber kode PHP dapat diubah dan didistribusikan secara bebas.

PHP (*Personal Home Page*) adalah pemrograman (*interpreter*) adalah proses penerjemahan baris sumber menjadi kode mesin yang dimengerti komputer secara langsung pada saat baris kode dijalankan (Sibero, 2012). Salah satu keunggulan PHP yaitu didukung oleh banyak *web server* seperti *apache*, *IIS*, *Lighttp* hingga *Xitami* dengan konfigurasi yang relatif mudah serta PHP bersifat open source yang dapat digunakan di berbagai sistem operasi (Linux, macintosh, windows).

2.10.3 *Cascading Style Sheets* (CSS)

CSS adalah suatu kumpulan kode-kode untuk memformat atau mengendalikan tampilan isi dalam suatu halaman web. Umumnya CSS digunakan untuk mengatur tampilan dokumen (Andi, 2013). CSS memungkinkan kita untuk menampilkan halaman yang sama dengan format yang berbeda. Selain itu dengan CSS, tampilan *website* akan lebih cantik dan konsisten.

Dalam pengkodean bahasa CSS, terdapat dua jenis yang dapat dipergunakan. Pertama secara internal dengan menuliskan langsung diantara *tag* HTML/XHTML. Kedua secara eksternal dengan

menuliskan kode CSS disimpan dalam *file* yang terpisah kemudian dipanggil saat halaman web dibuka, CSS merupakan sebuah teknologi internet yang direkomendasikan oleh W3C (*World Wide Web Consortium*) dan diperkenalkan pada tahun 1996.

2.10.4 *Hyper Text Markup Language* (HTML)

Hyper Text Markup Language adalah dokumen yang mengatur bahasa-bahasa yang digunakan untuk mendesain halaman web. HTML dirancang dengan menggunakan sistem markah/tanda (*tagging*) sehingga markah tersebut mampu ditampilkan menjadi halaman web utuh.

HTML (*Hyper Text Markup Language*) diuraikan sebagai bahasa standar yang digunakan oleh *browser* internet untuk membuat halaman dan dokumen pada sebuah web yang kemudian dapat diakses dan dibaca layaknya sebuah artikel.

2.10.5 *HyperText Transfer Protocol* (HTTP)

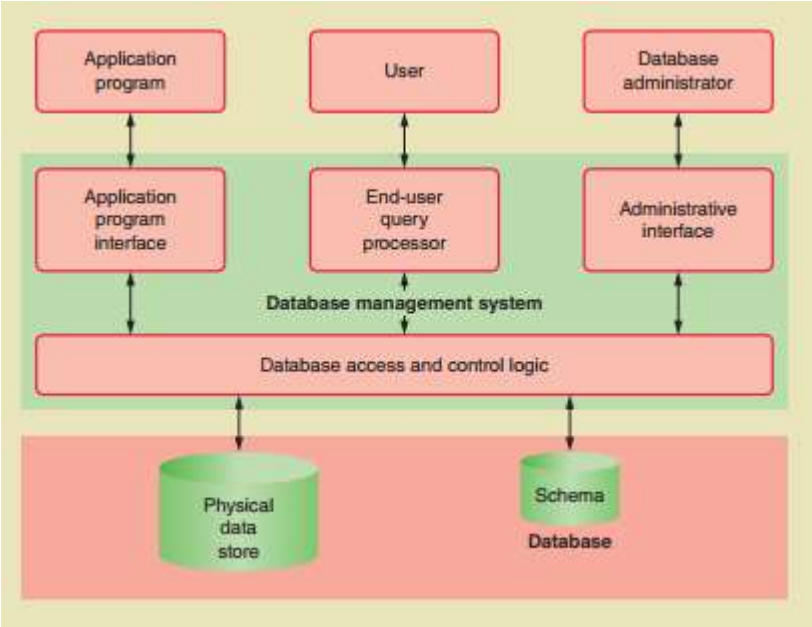
Hypertext Transfer Protocol (HTTP) merupakan sebuah protokol jaringan yang digunakan untuk sistem informasi yang bersifat terdistribusi, kolaboratif, dan menggunakan hipermedia banyak dalam memanfaatkan sumber daya yang dihubungkan dengan link yang disebut dokumen *hypertext* yang membentuk *World Wide Web*.

2.11 Basis Data (*Database*)

Database adalah kumpulan data yang terintegrasi dan disimpan terpusat dikelola serta dikendalikan (Satzinger et al, 2012). *Database* biasanya menyimpan banyak informasi. *database* merupakan sekumpulan data yang saling berkaitan dan berhubungan satu dengan yang lain, tersimpan di perangkat keras komputer dan menggunakan perangkat lunak untuk memanipulasinya. *Database* diakses atau dimanipulasi menggunakan perangkat lunak paket yang disebut DBMS (*Database Management System*).

2.11.1 Pengertian *Database Management System*

Database management system adalah sistem perangkat lunak yang memungkinkan pengguna dapat mendefinisikan, membuat, merawat, dan mengatur akses ke database (Satzinger et al, 2012). *Database management system* memiliki empat komponen utama: antarmuka program aplikasi (API), antarmuka kueri, antarmuka administratif, dan kumpulan data yang mendasari mengakses *program* dan *subroutines*.



Gambar 2.15 Komponen *Database Management System* (Satzinger et al, 2012)

2.11.2 *SQL Server Management*

MySQL adalah sebuah basis data yang dapat digunakan baik secara client maupun server. *Mysql* menggunakan format dasar *Standard Query Language* (SQL) (Peranginangin, 2006). *Mysql* dapat dijalankan di beberapa Sistem Operasi dan Bahasa Pemrograman. Salah satu bahasa pemrograman yang banyak menggunakan *Mysql* sebagai *database* adalah PHP.

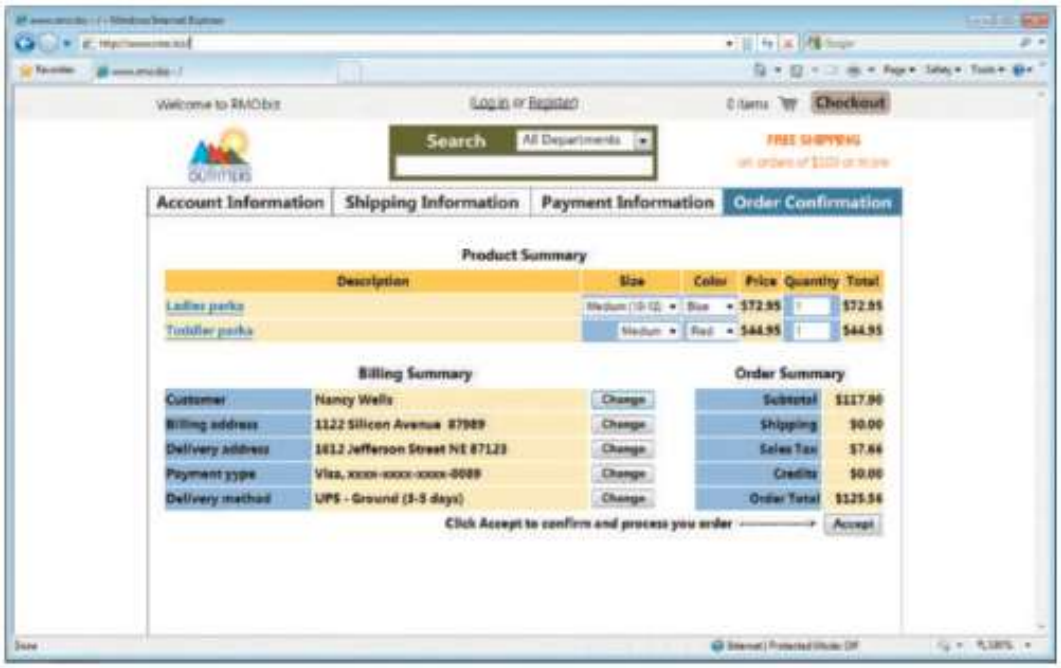
MySQL memiliki dua bentuk lisensi, yaitu *free software* dan *shareware*. *MySQL* yang banyak digunakan adalah *MySQL free software* yang berada di bawah lisensi GNU/GPL (*General Public Licence*).

MySQL memiliki keuntungan antara lain (Welling & Thomson, 2013):

a. MySQL memiliki performa yang cepat.
b. MySQL dapat digunakan secara gratis apabila menggunakan lisensi open source.
c. MySQL mudah digunakan dan dipelajari.
d. MySQL dapat digunakan di berbagai sistem operasi.
e. MySQL memungkinkan untuk mengubah source code.
f. MySQL mendukung penggunanya dengan training, konsultasi dan sertifikasi.

2.12 *System Interface*

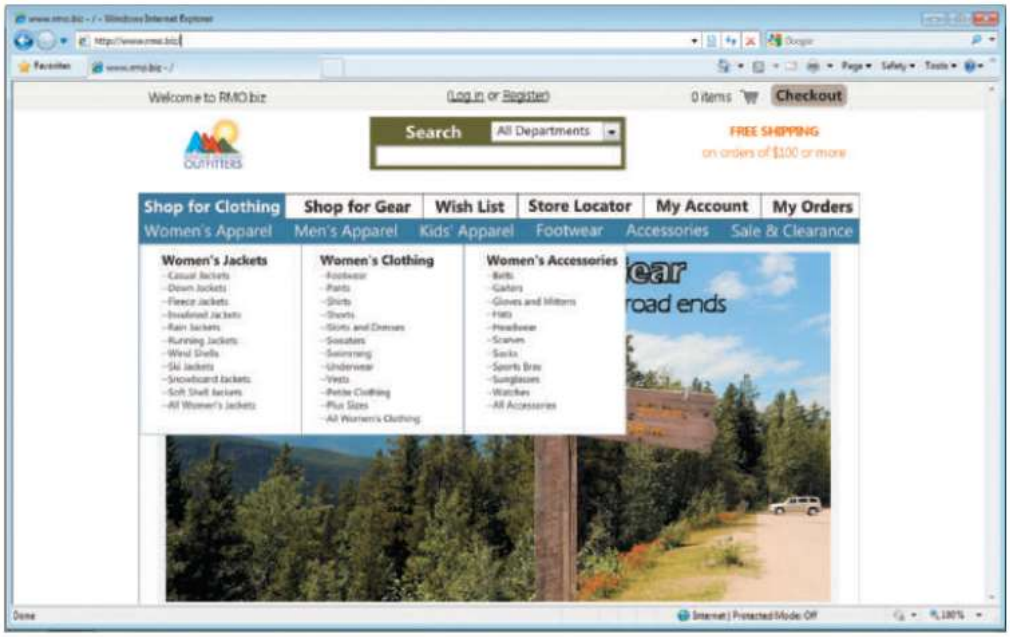
System Interface adalah *input* dan *output* yang membutuhkan intervensi manusia. Input ditangkap secara otomatis oleh perangkat *input* khusus seperti pemindai, pesan elektronik ke atau dari sistem lain, atau transaksi yang diambil oleh sistem lain. *Output* dianggap sebagai antarmuka sistem jika mereka mengirim pesan atau informasi ke sistem lain (Satzinger et al, 2012).



Gambar 2.16 Contoh *System interface website* (Satzinger et al, 2012)

2.13 *User Interface*

User interface adalah bagian dari sebuah sistem informasi yang membutuhkan interaksi pengguna untuk membuat *input* dan *output* (Satzinger et al, 2012).



Gambar 2.17 Contoh *User interface* website (Satzinger et al, 2012)

Terdapat delapan aturan emas (*eight golden rules*) yang merupakan aturan untuk merancang layar antarmuka yang interaktif dan mendukung fungsi kegunaan.

1. *Affordance and Visibility*

   fungsi menu–menu dalam sistem harus jelas dan kelihatan oleh pengguna serta dapat digunakan secara maksimal fungsi sistem tersebut.

2. *Consistency*

   Pengaturan informasi yang ada dalam sistem harus konsisten dan spesifik fungsi dari sistem harus jelas.

3. *Shortcut*

   kecepatan dalam mengakses informasi sangat penting dalam memudahkan pengguna.

4. *Feedback*

   Umpan balik harus dikirimkan kepada pengguna berupa informasi sesuai dengan aksi yang dilakukan oleh pengguna.

5. *Dialogs That Yield Closure*

   menyampaikan bahwa proses yang dijalankan oleh pengguna sudah selesai, pengguna paham bahwa tidak perlu menunggu apakah masih akan ada tahapan lain setelah menyelesaikan suatu proses.

6. *Error Handling*

   Sistem yang dirancang mampu mencegah kesalahan yang dilakukan oleh pengguna, apabila kesalahan terjadi sistem siap memberikan mekanisme penanganan.

7. *Easy Reversal of Actions*

   Pengguna merasa nyaman saat mencoba untuk melakukan eksplorasi pada aplikasi karena apabila ada kesalahan pengguna bisa memilih menu *back* atau *undo* untuk kembali.

8. *Reduce Short-Term Memory Load*

   pengguna tidak perlu mengingat data yang harus diinput ke sistem. Karena data yang harus diinput, sudah disediakan oleh sistem.

2.14 Kerangka Pemikiran

Berikut merupakan tahapan penelitian pada analisis dan perancangan sistem informasi *gift funding wedding* pada *Wemary*. Tahapan penelitian ditunjukan pada Gambar 2.18.

Gambar 2.18 Kerangka Pemikiran

1. Identifikasi Masalah

   Pada tahap Identifikasi Masalah ini, dilakukan proses pengidentifikasian masalah yang terjadi di dalam proses bisnis *Wemary* dengan cara mengevaluasi sistem yang saat ini sedang berjalan.

2. Studi Pustaka

   Analisa seputar industri terkait dari buku, jurnal, dan sumber terpercaya lainnya guna mendapatkan informasi yang mendukung untuk digunakan dalam perancangan sistem maupun penulisan penelitian.

3. Kuesioner

   Pada tahapan ini penulis melakukan pengumpulan data dengan metode kuesioner terbuka kepada beberapa responden yang mewakili tamu undangan dan calon pengantin.

4. Observasi

   Pada tahapan ini melakukan observasi terhadap proses bisnis yang berjalan pada *Wemary*.

5. Wawancara

   Dalam proses ini penulis melakukan wawancara dengan beberapa *stakeholder* dari *Wemary* yang terlibat dalam perancangan sistem. Wawancara dilakukan untuk mendapatkan informasi kebutuhan pengguna.

6. Metode *Waterfall* dan *Systems development life cycle* (SDLC)

   Dalam penelitian ini penulis menggunakan metode *waterfall* dalam perancangan sistem yang dibuat. Rangkaian proses yang dilakukan dalam penelitian ini mengacu pada SDLC. Tahapan yang dilakukan yaitu, tahap perencanaan (*planning*), tahap analisis (*analysis*), tahap perancangan (*design*).

7. UML Diagram

   Pada tahap ini peneliti membuat UML diagram yang terdiri dari *activity diagram, use case diagram, class diagram, sequence diagram* sebagai gambaran visual dari sistem yang nantinya akan dibangun sesuai dengan hasil dari tahapan sebelumnya.

8. Perancangan Sistem

   Pada tahap ini peneliti membuat perancangan sistem sesuai dengan hasil analisis dari tahapan yang telah dilalui. Perancangan sistem berupa membuat model purwarupa (*prototype*).

9. Kesimpulan dan Saran

   Pada tahap ini didapatkan kesimpulan bahwa sistem informasi yang dibangun nantinya akan dapat membantu pengantin dalam mewujudkan keinginannya. Serta saran untuk penelitian selanjutnya.