BAB 2
LANDASAN TEORI

2.1 Teori Umum

2.1.1 Sistem Informasi

Menurut Satzinger, Jackson, dan Burd (2012:5), sistem adalah sekumpulan kegiatan–kegiatan yang memungkinkan pengguna untuk mendefiniskan dan mendeskripsikan secara jelas di dalam memecahkan suatu permasalahan atas kebutuhan pengguna sistem tersebut.

Menurut pendapat O'Brien & Marakas (2010:4), informasi adalah data yang telah diubah menjadi isi yang mempunyai arti dan kegunaan untuk pengguna akhir tertentu. Dengan demikian informasi adalah data–data yang disimpan, diolah dan kemudian menghasilkan suatu pengetahuan atau keterangan yang berguna bagi para pembaca atau orang yang membutuhkan informasi tersebut.

Menurut Satzinger, Jackson, dan Burd (2012:5), sistem informasi merupakan sekumpulan komponen komputer yang terkait yang dikumpulkan, diproses, disimpan dan menyediakan output berupa informasi yang dibutuhkan untuk menyelesaikan tugas bisnis.

Sistem informasi telah didefinisikan dalam dua perspektif yaitu yang berkaitan dengan fungsinya dan yang lain terkait dengan strukturnya. Dari perspektif fungsional, sistem informasi adalah media yang diimplementasikan secara teknologi untuk tujuan merekam, menyimpan, dan menyebarluaskan ekspresi linguistik serta untuk mendukung pembuatan inferensi. Dari perspektif struktural, sistem informasi terdiri dari kumpulan orang, proses, data, model, teknologi dan bahasa yang diformalkan sebagian, membentuk struktur kohesif yang melayani beberapa tujuan atau fungsi organisasi.

2.1.2 Sistem Informasi Manajemen Rumah Sakit

Menurut Ery Rustiyanto (2010), Sistem Informasi Manajemen Rumah Sakit (SIM RS) adalah suatu rangkaian kegiatan yang mencakup semua pelayanan kesehatan (rumah sakit) di semua tingkatan administrasi yang dapat memberikan informasi kepada pengelola untuk proses manajemen pelayanan kesehatan di rumah sakit. Pelayanan yang

termasuk didalamnya adalah Pelayanan Utama (*Front Office*) dan Pelayanan Administrasi (*Back Office*).

- Pelayanan Utama (*Front Office*)

  Setiap Rumah Sakit memiliki prosedur yang unik (berbeda satu dengan lainnya), tetapi secara umum/generik memiliki prosedur pelayanan terintegrasi yang sama yaitu proses pendaftaran, proses rawat (jalan atau inap) dan proses pulang.

- Pelayanan Administrasi (*Back Office*)

  Rumah Sakit merupakan unit yang mengelola sumber daya fisik (manusia, uang, mesin/alat kesehatan/aset, material seperti obat, alat tulis kantor, barang habis pakai dan sejenisnya). Walaupun proses bisnis setiap Rumah Sakit unik tapi tetap terdapat proses umum, diantaranya perencanaan, pembelian/pengadaan, pemeliharaan stok/inventory, pengelolaan aset, pengelolaan SDM, pengelolaan uang (hutang, piutang, kas, buku besar dan lainnya).

Peran sistem informasi di dalam kegiatan manajemen rumah sakit sangatlah membantu dan mempunyai peran yang sangat efektif dalam proses pelayanan kesehatan di rumah sakit, dengan sistem informasi seorang pemimpin rumah sakit dapat mengambil suatu kebijakan secara cepat, tepat, dan akurat berdasarkan informasi yang didapat dari pelayanan kesehatan di rumah sakit yang dipimpinnya.

2.1.3 Sistem Peresepan

Resep elektronik adalah generasi dari peresepan melalui proses penambahan data secara otomatis menggunakan perangkat lunak dan jaringan transmisi yang menghubungkan resep dengan apotek. Resep elektronik diharapkan dapat menggantikan resep manual, resep yang dicetak komputer dan *computer faxed prescription*. Untuk menjamin kerahasiaan informasi pasien, resep elektronik dikirmkan melalui jaringan yang aman dan tertutup yang hanya ditujukan untuk kepentingan peresepan (Kusumarini, Dwiprahasto & Wardani, 2011). Selain Itu juga peresepan elektronik juga dapat digunakan untuk mengurangi terjadinya kesalahan penanganan medis akibat kesalahan pembacaan resep dokter.

e-Recipe merupakan singkatan dari *electronic recipe*, dimana *e-Recipe* ini merupakan suatu perancangan sistem yang menyediakan proses bisnis mengenai penyediaan resep obat. *Electronic recipe* ini merupakan proses bisnis dimana penyediaan resep obat yang dilakukan oleh dokter dan diberikan ke bagian farmasi masih menggunakan metode manual dan diubah dengan menggunakan teknologi agar lebih digital. Dengan menggunakan perancangan sistem ini, bagian farmasi tidak perlu memasukkan data pasien dan daftar obat yang telah di inputkan oleh dokter karena dengan menggunakan *scan barcode*, maka data pasien dan daftar obat tersebut akan langsung muncul di dalam sistem. Hal ini akan membantu dokter dan bagian farmasi untuk melakukan pekerjaan tersebut menjadi lebih cepat sehingga pasien pun tidak perlu untuk menunggu lama mendapatkan obat.

2.2 Teori Khusus

2.2.1 *Software Development Life Cycle* (SDLC)

Menurut Rainer, R. Kelly (2013:420), *System Development Life Cycle* (SDLC) adalah metode pengembangan sistem tradisional yang digunakan organisasi untuk proyek-proyek IT berskala besar. SDLC adalah kerangka terstruktur yang merupakan proses sekuensial di mana sistem informasi dikembangkan.

Menurut Satzinger, Jackson, dan Burd (2012:6), bahwa *System Development Life Cycle* (SDLC) mengidentifikasi semua kegiatan yang diperlukan untuk membangun, meluncurkan, dan memelihara suatu sistem informasi. 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 berhasil meluncurkan dan menyebarkan sistem informasi baru. Ada serangkaian proses inti yang selalu diperlukan, berikut adalah enam tahapan inti yang diperlukan :

1. Identifikasi masalah atau kebutuhan

   Pada tahapan ini dapat dilakukan mengidentifikasi permasalahan yang ada serta kebutuhannya. Setelah mendapatkan permasalahan dan kebutuhannya maka dapatkan persetujuan untuk melanjutkan tahapan.

2. Rencanakan dan pantau proyek

   Pada tahapan ini rencanakan mengenai apa yang harus dilakukan, bagaimana melakukannya dan siapa yang melakukannya.

3. Analisis *detail* masalah atau kebutuhan

   Setelah mengetahui masalah yang ada dan kebutuhan yang diperlukan maka selanjutnya temukan dan pahami *detail* masalah atau kebutuhannya tersebut.

4. Desain

   Membuat desain komponen-komponen sistem yang dapat memecahkan masalah atau memenuhi kebutuhan.

5. Implementasi

   Setelah desain komponen sistem sudah dibuat maka tahapan selanjutnya yaitu implementasi dengan pembangunan sistem. Sistem ini dibangun dan diuji untuk memastikan bahwa sistem melakukan sesuai yang dirancang. Pengujian adalah salah satu langkah paling penting dalam implementasi.

6. *Publish* sistem

   Sistem sudah di bangun dan sudah dilakukan pengujian, maka sistem akan di *publish* pada *production environment*.

Namun pada perancangan sistem *e-Recipe* ini hanya sampai perancangan saja maka dari itu tahapan-tahapan yang digunakan yaitu identifikasi masalah atau kebutuhan, rencanakan dan pantau proyek, analisis *detail* masalah atau kebutuhan, dan desain. Dengan tahapan-tahapan tersebut sudah memenuhi kebutuhan perancangan sistem *e-Recipe*.

2.2.2 *Unified Modeling Language* (UML)

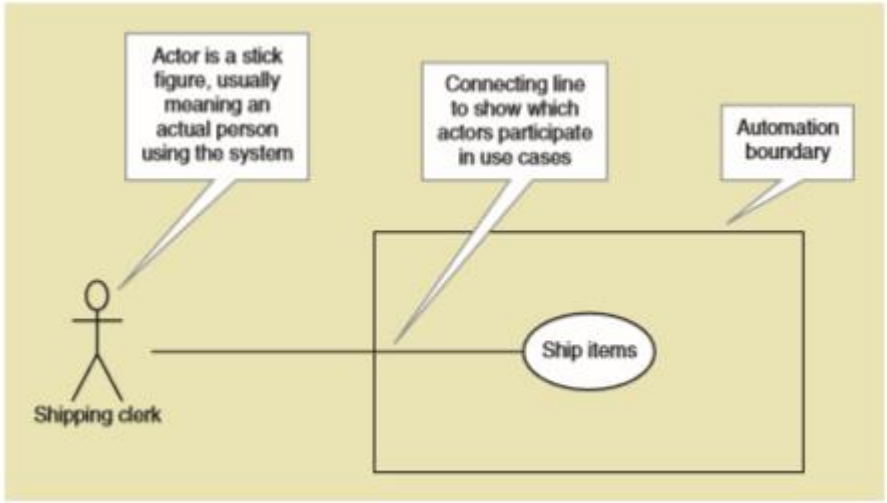
Menurut Satzinger, Jackson, dan Burd (2012:46), UML adalah seperangkat standar model dan notasi yang didefinisikan oleh *Object Management Group* (OMG) yang merupakan sebuah organisasi standar untuk pengembangan sistem. UML merupakan seperangkat model kontruksi dan notasi yang dibentuk dalam pengembangan sistem berorientasi pada objek.

UML adalah alat untuk menentukan dan memvisualisasikan sistem perangkat lunak. Ini termasuk jenis diagram standar yang

menggambarkan dan memetakan secara visual aplikasi komputer atau desain dan struktur sistem database. Penggunaan UML sebagai alat untuk mendefinisikan struktur suatu sistem adalah cara yang sangat berguna untuk mengelola sistem yang besar dan kompleks. Memiliki struktur yang terlihat jelas membuatnya mudah untuk memperkenalkan orang baru ke proyek yang ada. UML digunakan untuk menentukan, memvisualisasikan, memodifikasi, membangun dan mendokumentasikan artefak dari sistem perangkat lunak yang berorientasi objek yang sedang dikembangkan.

1. Use Case Diagram

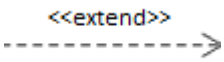
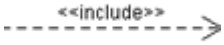
Sebuah *use case diagram* adalah model grafis yang merangkum informasi tentang aktor dan *use case* (Satzinger *et al*, 2012). *Use case* digunakan untuk mengetahui proses-proses apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi itu. *Use case diagram* dapat dilihat pada gambar 2.1 dibawah ini :



Gambar 2.1 *Use Case Diagram*
(Sumber: Satzinger, Jackson, dan Burd, 2012)

Untuk lebih jelasnya, simbol-simbol di atas dapat diuraikan ke dalam tabel di bawah ini :

Tabel 2.1 Simbol *Use Case Diagram*
(Sumber: Satzinger, Jackson, dan Burd, 2012)

No.	Simbol	Deskripsi
1.	*Use case* 	Aktivitas yang dapat dilakukan oleh aktor pada sistem
2.	Aktor / *actor* 	Merepresentasikan pelaku dari sistem.
3.	Asosiasi / *association*	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
4.	*Extends Relationship* 	Hubungan antar *use case* yang menunjukan bahwa satu *use case* memiliki kemungkinan untuk dilanjutkan ke *use case* lain.
5.	*Subject/Automation Boundary* 	Batasan otomasi/pelakunya bukan manusia.
6.	*Include Relationship* 	Hubungan antar *use case* yang menunjukan bahwa satu *use case* termasuk dalam *use case* lain.

2. Use Case Description

Menurut Satzinger, Jackson, dan Burd (2012:121), *Use Case Description* merupakan penjelasan *detail* 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 terperinci tentang setiap

use case dijelaskan dengan menggunakan deskripsi kasus. Ada tiga level pendeskripsian dari *use case*, yaitu *brief description, intermediate description,* dan *fully developed description. Use case description* dibedakan menjadi tiga, yaitu :

a. *Brief Description*

Penggunaan *brief description* biasanya digunakan untuk kasus penggunaan yang sangat sederhana, terutama ketika sistem yang akan dikembangkan adalah aplikasi kecil yang dipahami dengan baik. *Brief Description* dapat dikatakan sebagai ringkasan tentang apa yang dilakukan oleh sistem untuk merespon aksi dari pengguna.

b. *Intermediate Description*

Penggunaan *Intermediate description* dilakukan untuk deskripsi yang lebih *detail* dan merupakan perluasan dari sebuah *Brief Description*. Deskripsi ini memasukkan arus aktivitas-aktivitas internal untuk suatu *use case*. Jika terjadi *multiple scenarios* pada *use case*, maka tiap arus aktivitas dideskripsikan secara masing-masing.

c. *Fully Developed Description*

Fully Developed Description merupakan metode mendokumentasikan sebuah *use case* secara rapi atau formal. Penggunaan deskripsi ini pengguna dapat mengetahui secara *detail* dan mudah. Dimana pengguna benar-benar memahami proses bisnis dan cara sistem. *Fully Developed Description* berfungsi sebagai template standar untuk mendokumentasikan deskripsi yang dikembangkan sepenuhnya untuk *use case* dan skenario lainnya. Berikut penjelasan *fully developed use case* description dengan secara rinci :

a. *Use case name* digunakan untuk menjelaskan nama *use case.*
b. *Scenario* digunakan untuk menjelaskan *scenario* yang ada pada *use case* tersebut.
c. *Triggering event* digunakan untuk menjelaskan *event* apa yang memicu terjadinya *use case*.

d. *Brief description* digunakan untuk menjelaskan *brief description* atau deskripsi singkat akan *use case* tersebut.
e. *Actors* digunakan untuk menjelaskan *actor* yang terlibat dalam *use case.*
f. *Related use cases* digunakan untuk menjelaskan *use case* lain yang terlibat dengan *use case*.
g. *Stakeholders* digunakan untuk menjelaskan *stakeholder* yang terlibat dalam *use case*.
h. *Preconditions* menjelaskan keadaan apa yang harus dipenuhi sebelum *use case* dilakukan.
i. *Postconditions* menjelaskan keadaan apa yang harus dipenuhi setelah *use case* dijalankan.
j. *Flow of activities* digunakan untuk menjelaskan alur aktivitas antara *actor* dan sistem.
k. *Exception conditions* digunakan untuk menjelaskan *exception conditions*, yaitu keadaan alternatif yang mungkin terjadi.

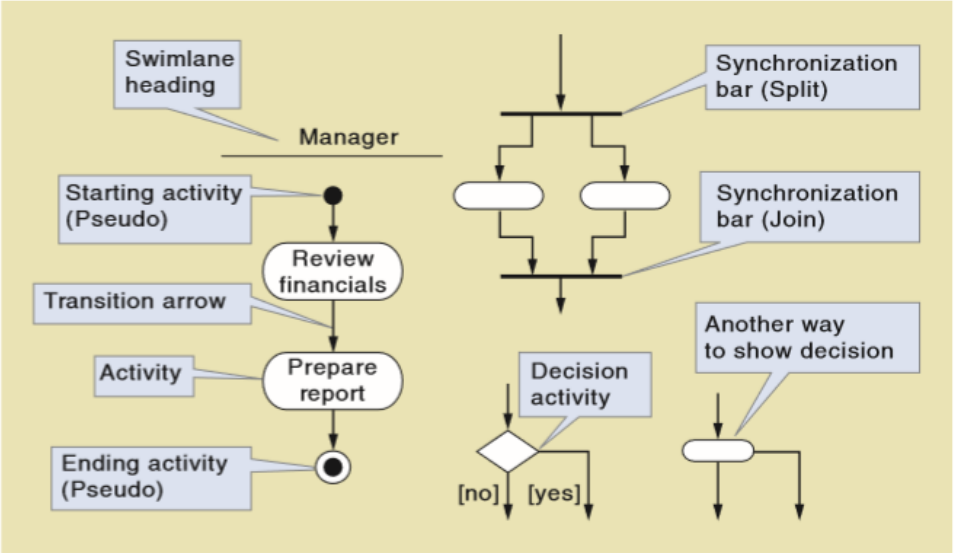
Berikut adalah contoh sebuah *Fully Developed Use Case Description* yang menggambarkan *use case* ship items:

Use case name:	*Ship items.*	
Scenario:	Ship items for a new sale.	
Triggering event:	Shipping is notified of a new sale to be shipped.	
Brief description:	Shipping retrieves sale details, finds each item and records it is shipped, records which items are not available, and sends shipment.	
Actors:	Shipping clerk.	
Related use cases	None.	
Stakeholders:	Sales, Marketing, Shipping, warehouse manager.	
Preconditions:	Customer and address must exist. Sale must exist. Sale items must exist.	
Postconditions:	Shipment is created and associated with shipper. Shipped sale items are updated as shipped and associated with the shipment. Unshipped items are marked as on back order. Shipping label is verified and produced.	
Flow of activities:	**Actor**	**System**
	1. Shipping requests sale and sale item information.	1.1 System looks up sale and returns customer, address, sale, and sales item information.
	2. Shipping assigns shipper.	2.1 System creates shipment and associates it with the shipper.
	3. For each available item, shipping records item is shipped.	3.1 System updates sale item as shipped and associates it with shipment.
	4. For each unavailable item, shipping records back order.	4.1 System updates sale item as on back order.
	5. Shipping requests shipping label supplying package size and weight.	5.1 System produces shipping label for shipment. 5.2 System records shipment cost.
Exception conditions:	2.1 Shipper is not available to that location, so select another. 3.1 If order item is damaged, get new item and updated item quantity. 3.1 If item bar code isn't scanning, shipping must enter bar code manually. 5.1 If printing label isn't printing correctly, the label must be addressed manually.	

Gambar 2.2 Contoh *Fully Developed Use Case Description*
(Sumber: Satzinger, Jackson, dan Burd, 2012)

3. Activity Diagram

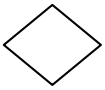
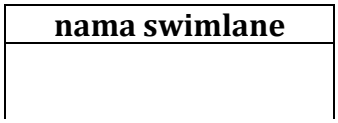
Sebuah *activity diagram* adalah sebuah diagram alir yang mendeskripsikan aktivitas dari beragam *user* atau *system*, orang yang melakukan tiap aktivitas, dan aliran sekuensial dari aktivitas tersebut (Satzinger *et al*, 2012:57). *Activity diagram* digunakan untuk menggambarkan bisnis dan alur kerja langkah demi langkah dari komponen dalam suatu sistem. *Activity diagram* juga menunjukkan keseluruhan aliran kontrol dalam sistem. Simbol-simbol yang biasa digunakan pada activity diagram dapat dilihat pada gambar 2.3 dibawah ini :

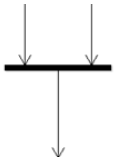
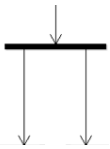


Gambar 2.3 Simbol *Activity Diagram*
(Sumber: Satzinger, Jackson, dan Burd, 2012)

Untuk lebih jelasnya, simbol-simbol di atas dapat diuraikan ke dalam tabel di bawah ini :

Tabel 2.2 Simbol *Activity Diagram*
(Sumber: Satzinger, Jackson, dan Burd, 2012)

No.	Simbol	Deskripsi
1.	*Initial* 	Inisialisasi/awal proses dari aktivitas sistem
2.	*Final* 	Akhir proses yang dilakukan sistem
3.	*Action* 	Menggambarkan suatu aktivitas
4.	Percabangan / *decision* 	Asosiasi percabangan dimana jika ada ada pilihan aktivitas lebih dari satu.
5.	Swimlane 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

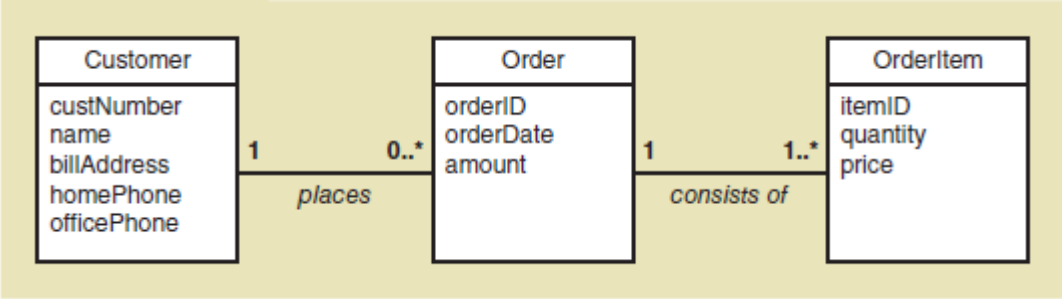
6.	Penggabungan / *join* 	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
7.	Percabangan/ *fork* 	Percabangan/pembagian proses ke dalam dua atau lebih jenis proses lanjutan yang berbeda

Berikut adalah contoh *activity diagram* yang menggambarkan proses *online checkout* :

Gambar 2.4 Contoh *Activity Diagram*
(Sumber: Satzinger, Jackson, dan Burd, 2012)

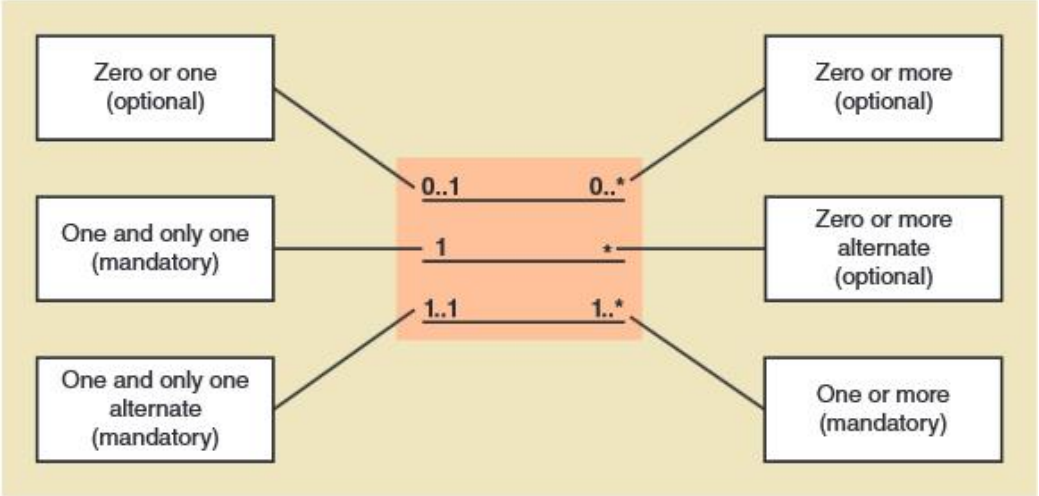
4. *Domain Class Diagram*

Domain Class Diagram menggambarkan identitas-identitas yang penting dalam menjalankan tugas para *user*, seperti *class-class problem domain*, hubungan antar *class-class* tersebut, dan atribut-atributnya. *Domain Model Class Diagram* adalah diagram yang digunakan untuk menggambarkan *class-class* yang terlibat berikut dengan keterkaitan atributnya (Satzinger *et al*, 2012:101). Contoh *domain class diagram* dapat dilihat pada gambar 2.5 dibawah ini :



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

n berikut adalah gambar 2.6 yang menjelaskan tentang *multiplicity* pada *class diagram* :

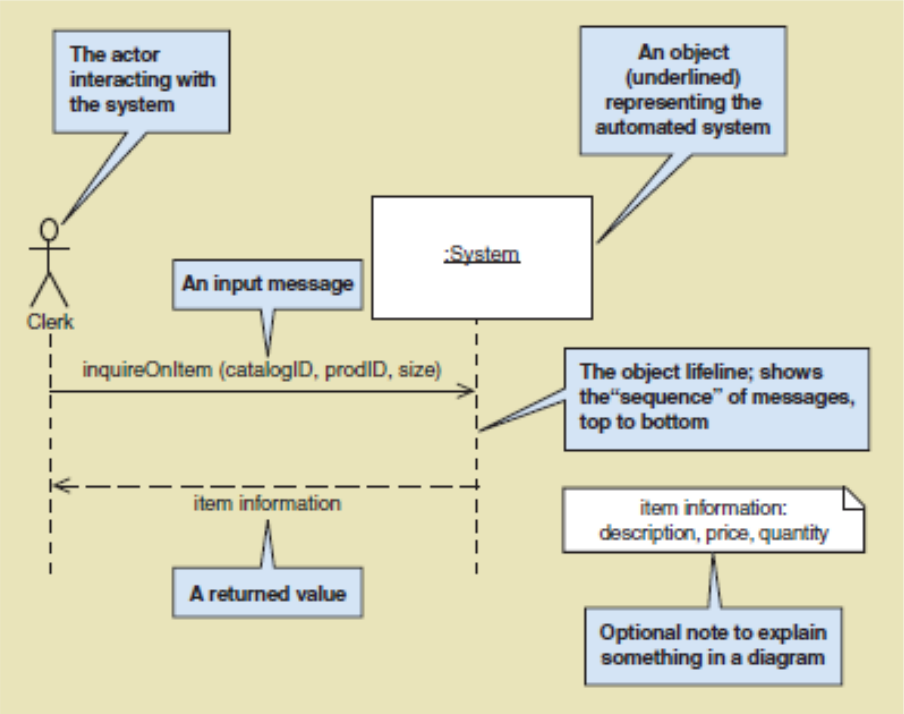


Gambar 2.6 *Multiplicity Class Diagram*
(Sumber: Satzinger, Jackson, dan Burd, 2012)

5. *System Sequence Diagram*

Menurut Satzinger, Jackson, dan Burd (2012:126), *System Sequence Diagram* merupakan diagram yang menunjukkan urutan pesan antara aktor eksternal dan internal sistem di dalam *use case*

atau *scenario* yang sudah dirancang sebelumnya. Contoh *system sequence diagram* dapat dilihat pada gambar 2.7 dibawah ini :



Gambar 2.7 Contoh *System Sequence Diagram*
(Sumber: Satzinger, Jackson, dan Burd, 2012)

System Sequence Diagram (SSD) terdiri dari beberapa notasi. Berikut merupakan penjelasan dari masing-masing notasi *system sequence diagram* :

a. *Lifeline or Object Lifeline*, yaitu garis di bawah objek pada SSD menunjukan kehidupan atau berlalunya waktu untuk suatu objek.
b. *Loof Frame*, yaitu notasi di dalam SSD yang menunjukan pengulangan pesan.
c. *True or False Condition*, yaitu bagian dari pesan diantara obyek yang dievaluasi sebelum ditransmisikan untuk menentukan apakah pesan dapat dikirim atau tidak.
d. *OptFrame*, yaitu notasi di dalam *sequence diagram* yang menunjukan pesan yang berupa optional atau pilihan.
e. *AltFrame*, yatu notasi di dalam *sequence diagram* yang menunjukan pesan *if-else*.

2.2.3 *Mock Up*

Mock up adalah representasi realistis atau gambaran secara *detail* dari produk misalnya *website* yang akan ditampilkan. *Mock up* menyampaikan aspek desain visual termasuk gambar, warna dan tipografi. *Mock up* sama dengan tampilan desain akhir dan terasa pada dasarnya. Memberikan gambaran tentang bagaimana desain akan terlihat seperti pixel ke pixel sebelum membuatnya hidup. *Mock up* berfokus pada pandangan produk, menambahkan unsur-unsur visual yang kaya dengan representasi visual berkualitas tinggi.

2.2.4 Star UML

Menurut Wikipedia dalam situsnya https://en.wikipedia.org/wiki/StarUML, StarUML adalah alat UML oleh MKLab. Perangkat lunak ini dilisensikan di bawah versi modifikasi GNU GPL hingga 2014, ketika versi yang ditulis ulang 2.0.0 dirilis untuk pengujian beta di bawah lisensi kepemilikan. StarUML mendukung sebagian besar jenis diagram yang ditentukan dalam UML 2.0. StarUML adalah platform pemodelan perangkat lunak yang mendukung UML (*Unified Modeling Language*). Ini secara aktif mendukung pendekatan MDA (*Model Driven Architecture*) dengan mendukung konsep profil UML. StarUML sangat bagus di sesuaikan ke lingkungan pengguna dan memiliki kemampuan diperpanjang yang tinggi dalam fungsinya. StarUML adalah platform pemodelan perangkat lunak yang dapat diperluas, tidak hanya menyediakan fungsi yang telah ditentukan, tetapi juga memungkinkan penambahan fungsi-fungsi baru.

2.2.5 Balsamiq

Menurut *websit*e resmi Balsamiq https://balsamiq.com/. Balsamiq *mock up* adalah program aplikasi yang digunakan dalam pembuatan tampilan *user interface* sebuah aplikasi. Software ini sudah menyediakan *tools* yang dapat memudahkan dalam membuat desain prototyping aplikasi yang akan kita buat. Software ini berfokus pada konten yang ingin digambar dan fungsionalitas yang dibutuhkan oleh pengguna. Alih-alih menggambar sketsa (*wireframe*) atau *prototype* rancangan desain *website* di atas kertas balsamiq *mock up* membantu seorang web desainer

membuat tampilan web dalam bentuk gambar di komputer. Tujuannya selain agar membuat tampilan (desain) *website* menarik juga dapat menyesuaikan dengan kebutuhan *customer* (pelanggan). Dengan alat pembuat *mock up* maka seorang web desainer dapat menganalisa tata letak, desain dan fungsi.

Kelebihan Balsamiq *Mock Up* dibanding software pembuat *mock up* lainnya adalah aplikasi ini berbasis *cloud*, disertai aplikasi desktop yang memungkinkan kita dengan cepat dan mudah membuat rancangan *website*. Dengan konten yang terbuat seperti dari gambaran tangan, akan membuat kita fokus pada pemecahan masalah *user interface* yang lebih besar, dari pada perincian *website*. Di websitenya sendiri ada dua pilihan untuk para pengguna, ada versi *trial for* dekstop dan ada juga yang bisa kita download untuk versi dekstop. Namun ada juga yang disediakan dalam versi berbayar. Aplikasi ini bisa digunakan untuk sistem operasi Windows, Mac OS, dan Linux.

2.3 *Literature Review*

Beberapa uraian tentang teori dari penelitian-penelitian yang sudah ada dan topiknya mirip dengan sistem perancangan *e-Recipe* dapat dilihat pada tabel di bawah ini :

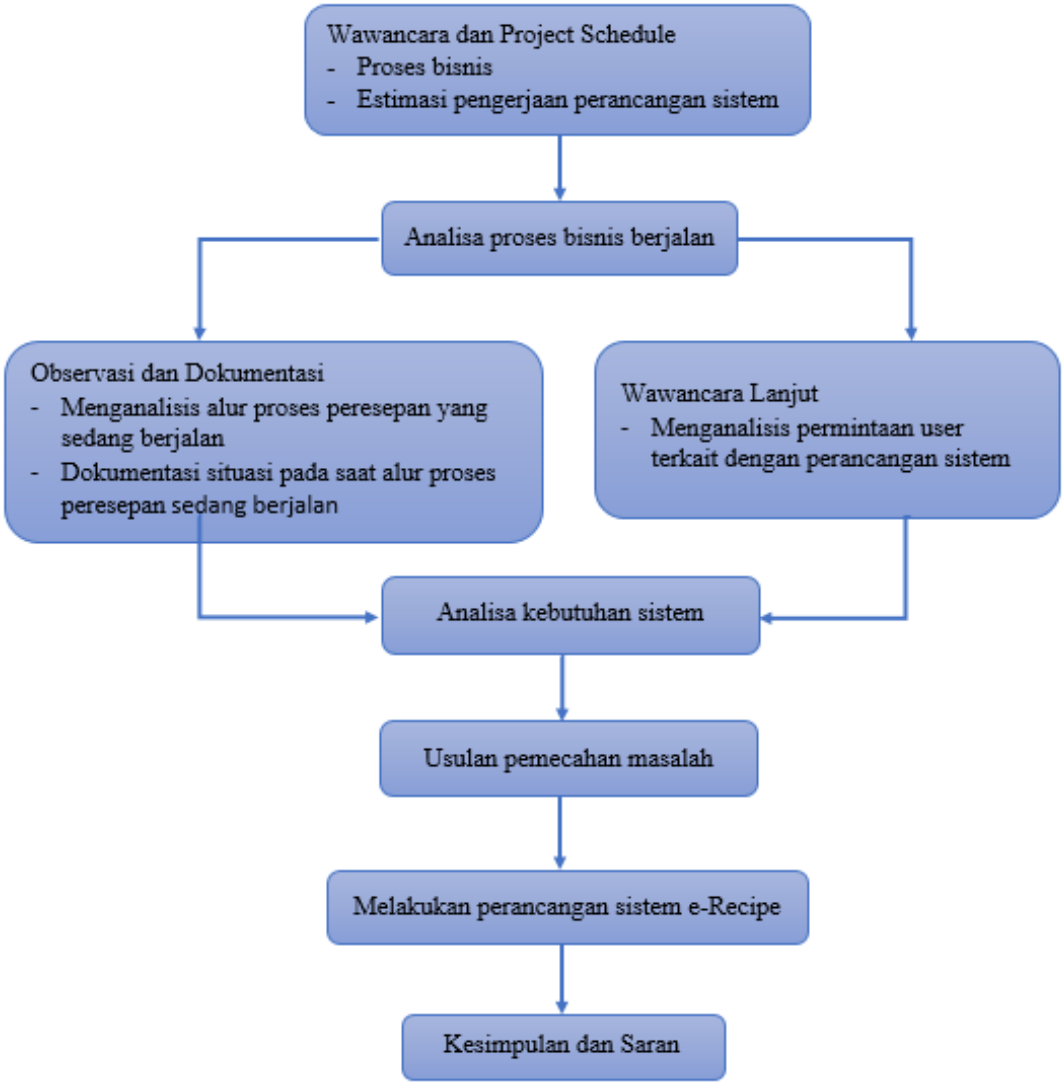
Tabel 2.3 *Literature Review*

Jurnal Nasional		
1.	Peneliti	Puspa Setia Pratiwi, Andri Lestari
	Judul	*E-Prescribing* : Studi Kasus Perancangan dan Implementasi Sistem Resep Obat Apotik Klinik
	Tahun	2013
	Hasil Penelitian	*E-Prescribing* dapat meningkatkan akses ke pelayanan kesehatan dan meningkatkan kualitas dan efektifitas dari pelayanan yang diberikan. Sistem *E-Prescribing* dapat mengurangi kesalahan pembuatan resep obat akibat resep yang ditulis dengan tangan. Dan dengan adanya sistem ini dapat menghemat waktu, tenaga, dan biaya dalam proses pemesanan, pembelian, dan pembuatan obat.

Jurnal Nasional		
2.	Peneliti	Nova Eka Diana, Dwi Agung Saputra
	Judul	*Expert E-Prescription Application* (EEPA) *Using* Forward Chaining Method
	Tahun	2015
	Hasil Penelitian	*Expert E-Prescription Application* (EEPA) dijadikan sebagai landasan untuk mengembangkan sistem pakar resep yang lebih detail dan akurat.
Jurnal Nasional		
3.	Peneliti	Sjamsul Arifin, Teduh Dirgahayu
	Judul	Evaluasi Implementasi Modul *E-Prescribing* Rumah Sakit Dengan Metode Pieces
	Tahun	2017
	Hasil Penelitian	*E-Prescribing* sudah diimplementasikan pada Rumah Sakit Sardjito Yogyakarta mempunyai sejumlah manfaat yaitu tidak terjadi risiko, salah baca, dosis obat tepat, input data cepat, irit kertas dan praktis. Selain itu mempermudah proses administrasi dan histori penggunaan obat oleh pasien.

2.4 Kerangka Pikir

Resep adalah permintaan tertulis dari seorang dokter kepada bagian farmasi yaitu apoteker. Dimana bagian farmasi akan menyiapkan atau membuat, meracik, serta menyerahkan obat kepada pasien. Sebelum bagian farmasi menyiapkan atau membuat, meracik, serta menyerahkan obat kepada pasien, bagian farmasi akan terlebih dahulu untuk membaca resep obat tersebut. Pembacaan resep dokter oleh bagian farmasi dapat mengakibatkan terjadinya kesalahan penanganan medis. Selain terjadinya kesalahan penanganan medis, mengakibatkan kendala lainnya. Maka dari itu untuk mengurangi kesalahan penanganan medis dan menangani kendala lainnya, perancangan sistem *e-Recipe* dibuat. Adapun bagan alur kerangka berpikir pada skripsi ini adalah sebagai berikut :



Gambar 2.8 Kerangka Pikir