BAB II
Landasan Teori

2.1 Tinjauan Pustaka

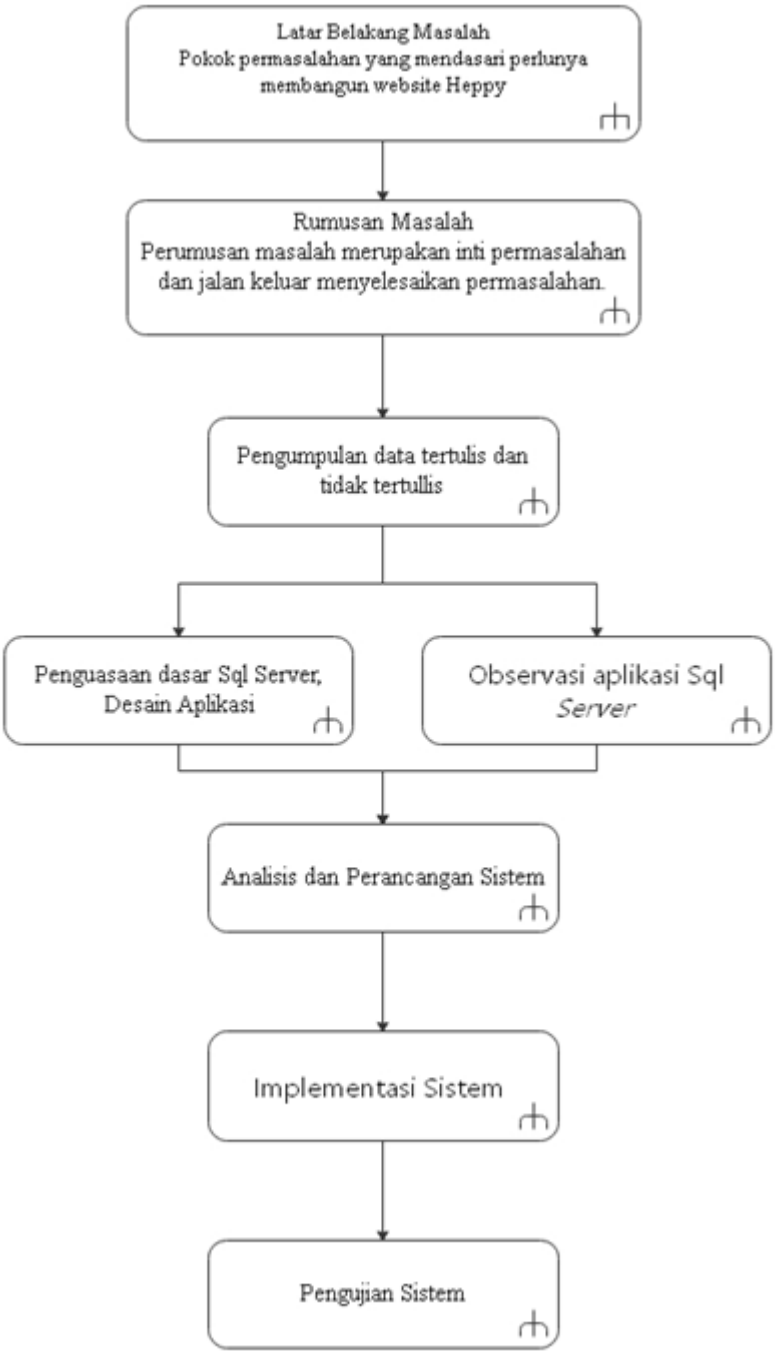
Untuk membangun sebuah sistem aplikasi yang baik agar sesuai dengan kebutuhan, maka diperlukan referensi yang lain seperti berikut:

(Kumlaasari, Retnoningsih, & Susilo, 2014) Aplikasi Penjualan Tiket Di Travel Jaya Sakti, dibangun menggunakan metode terstruktur. Sistem Informasi Pelayanan Tiket Di Travel Jaya Sakti merupakan program aplikasi yang diharapkan dapat membantu fungsi sistem pelayanan tiket khususnya pada Travel Jaya Sakti untuk mengatasi berbagai persoalan yang timbul akibat penggunaan sistem pencatatan manual (pencatatan pada buku). Berdasarkan hasil pengujian sistem dengan metode McCall yang telah dilakukan, sistem informasi ini mendapatkan nilai total kualitas 75,907%. Dari hasil pengujian tersebuti harapkan Aplikasi Penjualan Tiket ini dapat diterapkan pada Travel Jaya Sakti

(Riskiono, 2018) Adanya kehadiran teknologi informasi sangat membantu dalam menunjang kebutuhan manusia untuk mendapatkan informasi dalam waktu yang relatif cepat. Diamana teknologi saat ini sangat mendukung untuk memenuhi kebutuhan informasi yang cepat, yaitu salah satunya adalah dengan dikembangkanya sistem informasi pelayanan jasa *Tour* dan *Travel* pada *Smart Tour*. Dari permasalahan di atas maka dapat di sarikan terkait belum adanya sistem yang mampu mengatasi masalah dalam hal pemesanan tiket dan penyewaan mobil, sehingga masyarakat masih kesulitan untuk dapat melakukan pemesanan tiket dan penyewaan mobil secara *online*. Untuk itu diperlukan sistem informasi yang dapat menampilkan informasi pemesanan tiket dan penyewaan mobil secara *online*, sehingga informasi pemesanan tiket dan penyewaan mobil dapat dengan mudah diakses oleh masyarakat.

2.2 Kerangka Pemikiran

Uraian dari kerangka pemikiran penelitian tugas akhir pada Gambar 2.1 Kerangka pemikiran adalah sebagai berikut :



Gambar 2.1 Kerangka Pemikiran

Uraian dari kerangka berfikir sebagai berikut :

1) Latar belakang masalah

   Pokok permasalahan yang mendasari perlunya membangun *website* Heppy *Tour* & *Travel* untuk memudahkan proses penyebarluasan informasi ke publik.

2) Perumusan masalah

   Perumusan masalah merupakan inti permasalahan dan jalan keluar menyelesaikan permasalahan.

3) Pengumpulan data tertulis dan tidak tertulis

   Mengumpulkan data penelitian menggunakan metode observasi, dokumentasi, dan wawancara.

4) Penguasaan dasar

   Maksudnya adalah Pada penelitian dilakukan percobaan membuat sistem agar lebih menguasai, terlebihnya untuk bahasa pemrograman.

5) Observasi sistem

   Penelitian dilakukan pengamatan Pada sistem yang sebelumnya digunakan agar dapat menjadi referensi dalam penelitian ini.

6) Analisis dan perancangan sistem

   Menganalisis dan menrancang sistem yang akan di bangun sehingga sistem ini dapat mempermudah dalam pembuatan sistem.

7) Implementasi sistem

   Perlunya implementasi sistem ke dalam *website* yang telah di rancang dan di buat.

8) Pengujian sistem

   Perlu adanya pengujian sistem supaya mengetahui apakah dalam sistem tersebut ada yang kurang.

2.3 Teori Pendukung

2.3.1 Sistem

Sistem adalah sebuah tatanan (keterpaduan), sekelompok unsur atau elemen yang berhubungan satu dengan yang lain untuk mencapai suatu tujuan (Rahmawati, 2017).

2.3.2 *Website*

Website merupakan kumpulan halaman-halaman yang berisi informasi yang disimpan di internet yang bisa diakses atau dilihat melalui jaringan internet pada perangkat-perangkat yang bisa mengakses internet itu sendiri seperti komputer. Definisi kata *web* adalah *web* sebenarnya penyederhanaan dari sebuah istilah dalam dunia komputer yaitu *World Wide Web* yang merupakan bagian dari tekhnologi internet (Hastanti, 2015).

World Wide Web atau disingkat dengan nama www, merupakan sebuah sistem jaringan berbasis *Client-Server* yang mempergunakan protokol *HTTP (Hypertext Transfer Protocol)* dan *TCP/IP (Transmisson Control Protocol / Internet Protocol)* sebagai medianya. Karena kedua sistem ini mempunyai hubungan yang sangat erat, maka untuk saat ini sulit untuk membedakan antara *HTTP* dengan *WWW*.

Internet dapat diartikan sebagai jaringan komputer yang luas dan besar yang mendunia, yaitu menghubungkan pemakai komputer dari negara ke negara di seluruh dunia. Pada awalnya internet atau *web* hanya dipergunakan untuk kepentingan militer yaitu suatu teknologi yang dipergunakan untuk mengirimkan pesan melalui satelit. Akan tetapi lama kelamaan teknologi tersebut akhirnya meluas, dan bahkan internet pada saat ini sudah sama populernya dengan *telephone*. Informasi yang dikirimkan lewat internet dapat diakses ke seluruh dunia hanya dalam hitungan menit bahkan detik.

Teknologi yang digunakan menjadi sangat populer dan cepat sekali perkembangannya. Saat ini internet sudah tidak menjadi istilah yang asing lagi. Suatu informasi yang dikirimkan lewat internet dapat berupa teks, gambar maupun multimedia sehingga internet juga dimanfaatkan oleh perusahaan-perusahaan untuk mempromosikan produk-produknya dengan cepat dan mudah.

2.3.3 *PHP (Personal Home Page)*

PHP merupakan bahasa pemrograman pelengkap *HTML* (*Hypertext Markup Languange)* yang memungkinkan aplikasi *web* dinamis untuk pengolahan data, pemrosesan data dari *user* via *form*, membuat buku tamu, toko *online*, dan lain sebagainya, dengan mudah *PHP* dapat melakukan koneksi ke *database* karena *PHP* memang dilengkapi fitur yang memungkinkan koneksi ke *PHP* dilakukan dengan mudah, tanpa harus melakukan pemrograman yang memusingkan. *PHP* juga merupakan bahasa pemrograman berbasis *server side* yang dapat melakukan parsing *script PHP* menjadi *script web* sehingga dari sisi *client* menghasilkan suatu tampilan yang menarik.

Jadi dapat disimpulkan bahwa pengertian *Personal Home Page* (*PHP*) adalah bahasa pemrograman pelengkap *HTML* berbasis *server side* yang memungkinkan aplikasi *web* dinamis, dapat melakukan koneksi ke *database* dan menghasilkan suatu tampilan yang menarik (Rahmawati, 2017).

2.3.4 *HTML (Hypertext Markup Language)*

HTML adalah sebuah bahasa markah yang digunakan untuk membuat sebuah halaman *web*, menampilkan berbagai informasi di dalam sebuah penjelajah *web* internet dan pemformatan *hypertext* sederhana yang ditulis dalam berkas format *ASCII* agar dapat menghasilkan tampilan wujud yang terintegrasi.

HTML adalah bahasa *markup* untuk menstrukturkan dan menampilkan isi dari *World Wide Web*, sebuah teknologi inti dari internet.

Jadi dapat disimpulkan bahwa pengertian *Hypertext Markup Languange* (*HTML*) adalah sebuah bahasa *markup* untuk membuat sebuah halaman *web* yang menampilkan berbagai informasi untuk menstrukturkan dan menampilkan isi dari *World Wide Web* pada *browser* yang ditulis dalam berkas format *ASCII* (Rahmawati, 2017).

2.3.5 *MySQL*

Berbagai definisi tentang *MySQL* yang dikutip dari beberapa sumber yang dapat dilihat pada Tabel 2.1 untuk memberikan penjelasan lebih lanjut tentang *MySQL* (Rahmawati, 2017).

Tabel 2.1 Definisi *MySQL*

Sumber	Definisi
Faizal & Irnawati (2015:4)	*MySQL* adalah perangkat lunak sistem manajemen basis data *SQL* (bahasa Inggris : *database management system*) atau *DBMS* yang *multithread, multi-user*. *MySQL* menggunakan *SQL* (*Structure Query Language*) sebagai bahasa dasar untuk mengakses *database*.
Raharjo (2015:16)	*MySQL* merupakan *software RDBMS* (atau *server database*) yang dapat mengelola *database* dengan sangat cepat, dapat menampung data dalam jumlah sangat besar, dapat diakses oleh banyak user (*multi- user*) dan dapat melakukan suatu proses secara sinkron atau berbarengan (*multi-threaded*).
Nugroho (2014: 31)	*MySQL* adalah *software* atau program aplikasi *database,* yaitu *software* yang dapat dipakai untuk menyimpan data berupa informasi, teks dan juga angka.
Ardhana (2014:46)	*MySQL* adalah sebuah perangkat lunak sistem manajemen basis data *SQL* (*database management system)* atau *DBMS* yang *multithread,* dan *multi- user.*

Dari berbagai uraian pada Tabel 2.1 dapat disimpulkan bahwa pengertian *MySQL* adalah sebuah perangkat lunak sistem manajemen basis data *SQL* atau *DBMS sofware* yang dapat dipakai untuk menyimpan data berupa informasi, teks dan juga angka.

2.3.6 *XAMPP*

Xampp merupakan paket PHP yang berbasis *Open Source* yang dikembangkan oleh sebuah komunitas *Open Source* . Penggunaan perangkat lunak *XAMPP* diawali dengan *install* paket *XAMPP* pada halaman resmi. Tersedia beberapa *update* yang dapat di-*download* sesuai dengan *platform* komputer pengguna. Setelah penginstalan selesai maka pengguna dapat memulai pemrograman dengan membuka *XAMPP Control Panel* terlebih dahulu untuk mengaktifkan *service* yang disediakan seperti : Apache, *MySQL, FileZilla, Mercury dan Tomcat* dengan mengklik *Action* : *Star* (Rahmawati, 2017).

2.3.7 *UML*

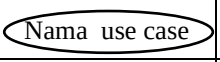
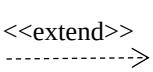
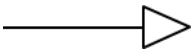
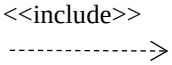
Pada perkembangan teknologi perangkat lunak diperlukan adanya bahasa yang digunakan untuk memodelkan perangkat lunak yang akan dibuat dan perlu adanya standarisari agar orang di berbagai negara dapat mengerti pemodelan perangkat lunak. Pada perkembangan teknik pemrograman berosientasi objek, muncullah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat

lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modelling Language* (Salahudin & Sukamto, 2016)

2.3.7.1 *Use Case Diagram*

Use case atau *diagram use case* merupakan pemodelan untuk melakukan sistem informasi yang akan dibuat. *Diagram use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. (Salahudin & Sukamto, 2016). Simbol-simbol yang ada pada diagram *use case* dapat dilihat pada Tabel 2.2.

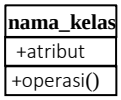
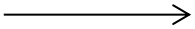
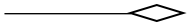
Tabel 2.2 Simbol-simbol *Use Case Diagram*

NO.	SIMBOL	NAMA	KETERANGAN
1.		*Use case*	Fungsional yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan kata kerja di awal di awal frase nama *use case.*
2.		*Actor*	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanyanya dinyatakan menggunakan kata benda di awal frase nama aktor.
3.		*Association*	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
4.		*Extend*	Relasi *use case* tambahan kesebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu.
5.		*Generalization*	Hubungan generalisasi dan spesialisasi (umum – khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
6.		*Include*	Relasi use case tambahan ke sebuah use case di mana use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya atau sebagai syarat dijalankan use case ini.

2.3.7.2 Class Diagram

Diagram kelas atau *class* diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas diagram memiliki atribut dan metode operasi (Salahudin & Sukamto, 2016). Simbol-simbol yang ada pada *class diagram* dapat dilihat pada Tabel 2.3.

Tabel 2.3 Simbol-simbol *Class Diagram.*

NO.	SIMBOL	NAMA	KETERANGAN
1.		*Class*	Kelas pada struktur sistem
2.		*Interface*	Sama dengan konsep interface dalam pemrograman berorientasi objek.
3.		*Association*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
4.		*Directed association*	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain biasanya juga disertai dengan *multiplicity.*
5.		*Generalization*	Relasi antar kelas dengan makna generalisasi - spesialisai (umum – khusus).
6.	---------->	*Dependency*	Relasi antar kelas dengan makna kebergantungan antar kelas.
7.		*aggregation*	Relasi antar kelas dengan makna semua bagian (*whole-part*).

2.3.7.3 Sequence Diagram

Diagram sekuen menggabarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek (Salahudin & Sukamto, 2016). Simbol-simbol yang ada pada *sequence* diagram dapat dilihat pada Tabel 2.4.

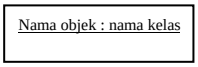
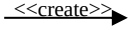
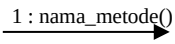
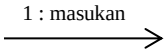
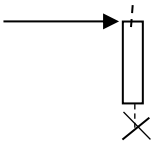
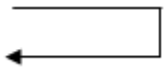
2.3.7.4 Activity Diagram

Diagram aktivitas atau *Activity diagram* menggambarkan *workflow* atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor (Salahudin & Sukamto, 2016). Simbol-simbol yang ada pada *activity* diagram dapat dilihat pada Tabel 2.5.

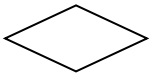
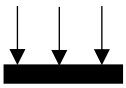
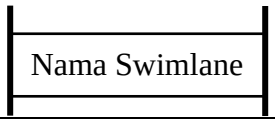
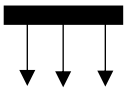
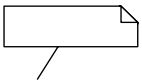
2.3.7.5 *Component Diagram*

Diagram komponen *atau component* diagram dibuat untuk menunjukkan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem (Salahudin & Sukamto, 2016). Simbol-simbol yang ada pada *component* diagram dapat dilihat pada Tabel 2.6.

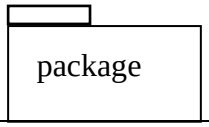
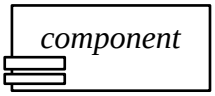
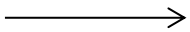
Tabel 2.4 Simbol-simbol *Sequence Diagram.*

NO.	SIMBOL	NAMA	KETERANGAN
1.		Aktor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanyanya dinyatakan menggunakan kata benda di awal frase nama aktor.
2.		*Lifeline*	Menyatakan kehidupan suatu objek.
3.		Objek	Menyatakan objek yang berinteraksi pesan.
4.		Waktu aktif	Menyatakan objek dalam keadaan aktif dan berinteraksi.
5.		Pesan tipe create	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat
6.		Pesan tipe call	Menyatakan suatu objek memanggil operasi / metode yang ada pada objek lain atau dirinya sendiri.
7.		Pesan tipe send	Menyatakan bahwa suatu objek mengirimkan data / masukan / informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi.
8.		Pesan tipe *destroy*	Menyatakan suatu objek mengakhiri hidup objek lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy.*
9		Pesan tipe *return*	Menyatakan bahwa suatu objek telah menjalankan suatu operasi atau metode menghasilkan suatu pengembailan ke objek tertentu, arah panah ke objek kembalian

Tabel 2.5 Simbol-simbol *Activity Diagram.*

NO	GAMBAR	NAMA	KETERANGAN
1		Status awal	Status awal aktivitas sistem.
2		Aktivitas	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3		*Decision*	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4		*Join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5.		Status akhir	Status akhir yang dilakukan sistem.
6.		*Swimlane*	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.
7.		*Fork*	Asosiasi pencabangan dimana lebih dari satu aktivitas cabangkan menjadi satu.
8.		*Note*	Note adalah diagram diagram yang tidak memiliki pengaruh semantik pada elemen model.

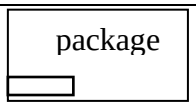
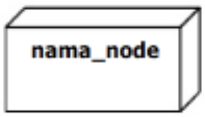
Tabel 2.6 Simbol – simbol *Component Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1.		*package*	*Package* merupakan sebuah bungkusan dari satu atau lebih komponen
2.		*Component*	Komponen sistem
3.		*Dependency*	Kebergantungan antar komponen, arah panah mengarah pada komponen yang dipakai
4.		*interface*	Sama dengan konsep *interface* pada pemrograman berorientasi objek, yaitu sebagai antarmuka komponen agar tidak mengakses langsung komponen
5.		*Link*	Relasi antar komponen

2.3.7.6 Deployment Diagram

Deployment diagram menunjukan konfigurasi komponen dalam proses eksekusi aplikasi. Diagram *deployment* juga dapat digunakan untuk memodelkan sistem tambahan, sistem *client*, sistem terdistribusi murni, rekayasa ulang aplikasi (Salahudin & Sukamto, 2016). Simbol-simbol yang ada pada *deployment* diagram dapat dilihat pada Tabel 2.7.

Tabel 2.7 Simbol-Simbol *Deployment Diagram*

NO	SIMBOL	NAMA	KETERANGAN
1.		*Package*	*Package* merupakan sebuah bungkusan dari satu atau lebih *node.*
2.		*Node*	Biasanya mengacu pada perangkat keras (*hardware*), perangkat lunak yang tidak dibuat sendiri (*software*), jika didalam *node* disertakan komponen untuk mengkonsistenkan rancangan maka komponen yang diikutsertakan harus sesuai dengan komponen yang telah didefinisikan sebelum pada diagram komponen.
3.		*Dependency*	Kebergantungan antar *node*, arah panah mengarah pada *node* yang dipakai.
4.		*Link*	Relasi antar *node.*

2.3.8 *Blackbox Testing*

Black-Box Testing (pengujian kotak hitam) yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi – fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan.

Pengujian kotak hitam dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan. Kasus uji yang dibuat untuk melakukan pengujian kotak hitam harus dibuat dengan benar dan kasus salah, misalkan untuk kasus proses login maka kasus uji yang dibuat adalah :

1. Jika *user* memasukkan nama pemakai (*username*) dan kata sandi (*password*) yang benar.

2. Jika *user* memasukkan nama pemakai (*username*) dan kata sandi (*password*) yang salah, misalnya nama pemakai benar tapi kata sandi salah, atau sebaliknya, atau keduanya salah (Winarno, dkk, 2014).

Pengujian *black-box* berfokus pada persyaratan fungsional perangkat lunak. Pengujian *black-box* memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi *input* yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Pengujian black-box bukan merupakan alternatif dari teknik *white-box,* tetapi merupakan pendekatan komplementer yang kemungkinan besar mampu mengungkap kelas kesalahan daripada metode *white-box.* Pengujian *black-box* berusaha menemukan kesalahan dalam kategori sebagai berikut: (1) fungsi-fungsi yang tidak benar atau hilang, (2) kesalahan *interface,* (3) kesalahan dalam struktur data atau akses *database external,* (4) kesalahan kinerja, (5) inisialisasi dan kesalahan terminasi

Tidak seperti pengujian *white-box* yang dilakukan pada saat awal proses pengujian, pengujian *black-box* cenderung diaplikasikan selama tahap akhir pengujian. Karena pengujian *black-box* memperhatikan struktur kontrol, maka perhatian berfokus pada *domain* informasi. Pengujian di deasin untuk menjawab pertanyaan-pertanyaan berikut :

1. Bagaimana validitas fungsional diuji ?
2. Kelas *ipnut* apa yang akan membuat *test case* menjadi baik ?
3. Apakah sistem sangat sensitif terhadap harga input tertentu ?
4. Bagaimana batasan dari suatu data diisolasi ?
5. Kecepatan data apa dan volume data apa yang dapat ditolelir oleh sistem ?
6. Apa pengaruh kombinasi tertentu dari data terhadap operasi sistem ?

Mengaplikasikan teknik *black-box,* maka serangkaian *test case* akan memenuhi kriteria berikut ini : (1) *test case* yang mengurangi, dengan harga lebih dari satu, jumlah *test case* tambahan yang harus didesain untuk mencapai pengujian yang dapat dipertanggungjawabkan, dan (2) *test case* yang memberi tahu mengenai kehadiran atau ketidakhadiran kelas kesalahan, daripada memberitahu kesalahan yang berhubungan hanya dengan pengujian spesifik yang ada (Pressman, 2002).