BAB II

TINJAUAN PUSTAKA

II.1. Sistem Penunjang Keputusan

Sistem pendukung keputusan merupakan sistem informasi interaktif yang menyediakan informasi, pemodelan dan memanipulasi data. Sistem itu digunakan untuk membantu pengambilan keputusan dalam situasi yang semiterstruktural dan situasi yang tidak terstruktur dimana tak seorang pun tahu sacara pasti bagaimana keputusan seharusnya dibuat. Sistem pendukung keputusan biasanya dibangun untuk mendukung solusi atas suatu masalah atau untuk mengevaluasi suatu peluang. Sistem pendukung keputusan yang seperti itu disebut aplikasi Sistem pendukung keputusan. Aplikasi Sistem pendukung keputusan digunakan dalam pengambilan keputusan. Aplikasi menggunakan CBIS *(Computer Based Information System)* yang fleksibel, interaktif, dan dapat diadaptasi,yang dikembangkan untuk mendukung solusi atas masalah manajemen spesifik yang tidak terstruktur. (Sylvia Hartati Saragih ; 2013 : 83)

Sistem pendukung keputusan tidak dimaksudkan unutk mengotomatisasikan pengambilan keputusan, tetapi memberikan perangkat interaktif yang memungkinkan pengambilan keputusan untuk melakukan berbagai analisis menggunakan model-model yang tersedia. Tujuan dari DSS adalah :

1. Membantu manajer dalam pengambilan keputusan atas masalah semistuktur.
2. Memberikan dukungan atas pertimbangan manajer dan bukannya dimaksudkan untuk menggantikan fungsi manajer.

3. Meningkatkan efektifitas keputusan yang diambil lebih daripada perbaikan efisiensinya.
4. Kecepatan komputasi. Komputer memungkinkan para pengambil keputusan untuk melakukan banyak komputansi secara cepat dengan biaya rendah.
5. Peningkatan produktivitas.
6. Dukungan kualitas.
7. Berdaya saing.
8. Mengatasi keterbatasan kognitif dalam pemrosesan dan penyimpanan.(*Sylvia Hartati Saragih ; 2013 : 83*)

Sistem Pendukung Keputusan merupakan suatu sistem yang mampu meningkatkan efektivitas dan efisiensi manajemen, meningkatkan kecepatan dan validitas pengambilan keputusan yang berkaitan dengan kegiatan operasional, dan meningkatkan kualitas SDM. Kecepatan dan validitas dalam mengolah informasi tersebut di atas merupakan syarat utama untuk mendukung keputusan, sehingga sistem pendukung keputusan yang digunakan juga harus memiliki perencanaan secara komprehensif dan terpadu untuk mengecilkan tingkat resiko kegagalan pengembangan dan pemilihan keputusan, untuk itu digunakan metode Analytic Hierarchy Process (AHP) yang mengkomposisi suatu masalah kompleks dan multikriteria kedalam suatu tatanan hierarki, yang mana dalam setiap tingkatan diisi dengan elemen tertentu. Metode penelitian yang digunakan dalam penelitian ini adalah penelitian rekayasa perangkat Lunak (software engineering) ,yaitu dengan metode Systems Development Life Cycle yaitu siklus hidup pengembangan system yang dimulai dari satu tahapan sampai tahapan akhir dan

kembali lagi ketahapan awal membentuk siklus atau daur hidup dalam penyelesaian ataupun pembuatan pengembangan suatu system informasi Hasil penelitian yang dilakukan penulis tidak hanya terfokus pada hasil perhitungan AHP tetapi juga memberikan perbandingan hasil terhadap pengambilan keputusan secara tepat. (*Tri Handoyo* ; 2013 : 377)

Adapun kesimpulan dari sistem penunjang keputusan adalah sebagai berikut :

a. Sistem yang dibangun dapat membantu manajemen dalam menyajikan sebuah informasi yang diperlukan sebagai sarana untuk mendukung suatu keputusan.
b. Proses pembuatan Sistem Pendukung Keputusan dapat dilakukan dengan metode Analytical Hiarachy Process dengan menentukan kriteria dan bobot untuk dihitung secara sistematik.

II.1.1. Hakikat Sistem Penunjang Keputusan

Pada dasarnya pengambilan keputusan adalah suatu pendekatan pendekatan sistematis pada hakekat suatu masalah, pengumpulan fakta-fakta, penentuan yang matang dari alternatif yang dihadapi, dan pengambilan tindakan yang menurut perhitungan merupakan tindakan yang paling tepat. Pada sisi lain, pembuat keputusan kerap kali dihadapkan pada kerumitan dan lingkup pengambilan keputusan dengan data yang begitu banyak. Untuk kepentingan itu, sebagian besar pembuat keputusan mempertimbangkan rasio biaya atau manfaat, dihadapkan pada suatu keharusan untuk mengandalkan seperangkat sistem yang mampu memecahkan masalah secara efisien dan efektif, yang kemudian disebut Sistem Pendukung Keputusan (SPK). (*Iwan Rijayana ; 2012 : C-49*)

II.1.2. *Analytical Hierarchy Process* (AHP)

Analytical Hierarchy Process (AHP) merupakan suatu model pendukung keputusan yang dikembangkan oleh *Thomas L. Saaty*. Model pendukung keputusan ini akan menguraikan masalah multi faktor atau multi kriteria yang kompleks menjadi suatu hirarki, menurut *Saaty*, hirarki didefinisikan sebagai suatu representasi dari sebuah permasalahan yang kompleks dalam suatu struktur multi level dimana level pertama adalah tujuan, yang diikuti level faktor, kriteria, sub kriteria, dan seterusnya ke bawah hingga level terakhir dari alternatif. Dengan hirarki, suatu masalah yang kompleks dapat diuraikan ke dalam kelompok-kelompoknya yang kemudian diatur menjadi suatu bentuk hirarki sehingga permasalahan akan tampak lebih terstruktur dan sistematis. sering digunakan sebagai metode pemecahan masalah dibanding dengan metode yang lain karena alasan-alasan sebagai berikut:

1. Struktur yang berhirarki, sebagai konsekuesi dari kriteria yang dipilih, sampai pada subkriteria yang paling dalam.
2. Memperhitungkan validitas sampai dengan batas toleransi inkonsistensi berbagai kriteria dan alternatif yang dipilih oleh pengambil keputusan. Memperhitungkan daya tahan *output* analisis sensitivitas pengambilan keputusan. (*Sylvia Hartati Saragih ; 2013 : 83-84*)

Analytical Hierarchy Process (AHP) merupakan salah satu metode untuk membantu menyusun suatu prioritas dari berbagai pilihan dengan menggunakan berbagai kriteria. Karena sifatnya yang multikriteria, AHP cukup banyak digunakan dalam penyusunan prioritas. Sebagai contoh untuk menyusun prioritas

penelitian, pihak manajemen lembaga penelitian sering menggunakan beberapa kriteria seperti dampak penelitian, biaya, kemampuan SDM, dan waktu pelaksanaan. Dalam menyelesaikan permasalahan dengan AHP ada beberapa prinsip yang harus dipahami diantaranya adalah sebagai berikut:

1. Membuat Hierarki

   Sistem yang kompleks bisa dipahami dengan memecahnya menjadi elemen-elemen pendukung, menyusun elemen secara hierarki, dan menggabungkannya atau mensistesisnya.

2. Penilaian kriteria dan alternatif

   Kriteria dan alternatif dilakukan dengan perbandingan berpasangan.

3. Synthesis of Priority (Penentuan Prioritas)

   Untuk setiap kriteria dan alternatif, perlu dilakukan perbandingan berpasangan (Pairwise Comparisons). Nilai-nilai perbandingan relatif dari seluruh alternatif kriteria bisa disesuaikan dengan judgement yang telah ditentukan untuk menghasilkan bobot dan prioritas. Bobot dan prioritas dihitung dengan memanipulasi matriks atau melalui penyelesaian persamaan matematika.

4. Logical Consistency (Konsistensi Logis)

   Konsistensi memiliki dua makna. Pertama, objek-objek yang serupa bisa dikelompokkan sesuai dengan keseragaman dan relevansi. Kedua, menyangkut tingkat hubungan antar objek yang didasarkan pada kriteria tertentu. Penghitungan konsistensi logis dilakukan dengan mengikuti langkah-langkah sebagai berikut :

   a. Mengalikan matriks dengan proritas bersesuaian.

b. Menjumlahkan hasil perkalian per baris.

c. Hasil penjumlahan tiap baris dibagi prioritas bersangkutan dan hasilnya dijumlahkan.

d. Hasil c dibagi jumlah elemen, akan didapat λmaks.

e. Indeks Konsistensi (CI) = (λmaks-n) / (n-1)

f. Rasio Konsistensi = CI/ RI, di mana RI adalah indeks random konsistensi. Jika rasio konsistensi ≤ 0.1, hasil perhitungan data dapat dibenarkan. (Tri Handoyo ; 2013 : 377-378)

Adapun kesimpulan dari *Analytical Hiarachy Process* adalah sebagai berikut :

1. Metode *Analytical Hiarachy Process* merupakan metode sistem Pendukung Keputusan yang bisa memecahkan berbagai masalah pengambilan keputusan multikriteria.
2. Sistem Pendukung Keputusan dengan model *Analytical Hiarachy Process* (AHP) memberikan manfaat kemudahan/ banyaknya alternatif pilihan keputusan.

II.1.3. Tahapan Metode AHP

Dalam metode *Analytical Hierarchy Process* dilakukan langkah-langkah sebagai berikut:

1. Mendefinisikan masalah dan menentukan solusi yang diinginkan.

   Dalam tahap ini penulis berusaha menentukan masalah yang akan penulis pecahkan secara jelas, detail dan mudah dipahami. Dari masalah yang ada penulis coba tentukan solusi yang mungkin cocok bagi masalah tersebut.

Solusi dari masalah mungkin berjumlah lebih dari satu. Solusi tersebut nantinya penulis kembangkan lebih lanjut dalam tahap berikutnya.

2. Membuat struktur hierarki yang diawali dengan tujuan utama.

   Setelah menyusun tujuan utama sebagai level teratas akan disusun level hirarki yang berada di bawahnya yaitu kriteria-kriteria yang cocok untuk mempertimbangkan atau menilai alternatif yang penulis berikan dan menentukan alternatif tersebut. Tiap kriteria mempunyai intensitas yang berbeda-beda. Hirarki dilanjutkan dengan subkriteria (jika mungkin diperlukan).

3. Membuat matrik perbandingan berpasangan yang menggambarkan kontribusi relatif atau pengaruh setiap elemen terhadap tujuan atau kriteria yang setingkat di atasnya. Matriks yang digunakan bersifat sederhana, memiliki kedudukan kuat untuk kerangka konsistensi, mendapatkan informasi lain yang mungkin dibutuhkan dengan semua perbandingan yang mungkin dan mampu menganalisis kepekaan prioritas secara keseluruhan untuk perubahan pertimbangan. Pendekatan dengan matriks mencerminkan aspek ganda dalam prioritas yaitu mendominasi dan didominasi. Perbandingan dilakukan berdasarkan *judgment* dari pengambil keputusan dengan menilai tingkat kepentingan suatu elemen dibandingkan elemen lainnya. Untuk memulai proses perbandingan berpasangan dipilih sebuah kriteria dari level paling atas hirarki misalnya K dan kemudian dari level di bawahnya diambil elemen yang akan dibandingkan misalnya E1,E2,E3,E4,E5.

4. Melakukan Mendefinisikan perbandingan berpasangan sehingga diperoleh jumlah penilaian seluruhnya sebanyak n x [(n-1)/2] buah, dengan n adalah banyaknya elemen yang dibandingkan. Hasil perbandingan dari masing-masing elemen akan berupa angka dari 1 sampai 9 yang menunjukkan perbandingan tingkat kepentingan suatu elemen. Apabila suatu elemen dalam matriks dibandingkan dengan dirinya sendiri

   maka hasil perbandingan diberi nilai 1. Skala 9 telah terbukti dapat diterima dan bisa membedakan intensitas antar elemen. Hasil perbandingan tersebut diisikan pada sel yang bersesuaian dengan elemen yang dibandingkan. Skala perbandingan perbandingan berpasangan dan maknanya yang diperkenalkan oleh Saaty bisa dilihat di bawah. Intensitas Kepentingan:

   a. 1 berarti kedua elemen sama pentingnya, Dua elemen mempunyai pengaruh yang sama besar
   b. 3 berarti elemen yang satu sedikit lebih penting daripada elemen yanga lainnya, Pengalaman dan penilaian sedikit menyokong satu elemen dibandingkan elemen yang lainnya
   c. 5 berarti elemen yang satu lebih penting daripada yang lainnya, Pengalaman dan penilaian sangat kuat menyokong satu elemen dibandingkan elemen yang lainnya
   d. 7 berarti satu elemen jelas lebih mutlak penting daripada elemen lainnya, Satu elemen yang kuat disokong dan dominan terlihat dalam praktek.

e. 9 berarti satu elemen mutlak penting daripada elemen lainnya, Bukti yang mendukung elemen yang satu terhadap elemen lain memiliki tingkat penegasan tertinggi yang mungkin menguatkan.

f. 2,4,6,8 berarti nilai-nilai antara dua nilai pertimbangan-pertimbangan yang berdekatan, Nilai ini diberikan bila ada dua kompromi di antara 2 pilihan Kebalikan = Jika untuk aktivitas i mendapat satu angka dibanding dengan aktivitas j , maka j mempunyai nilai kebalikannya dibanding dengan i

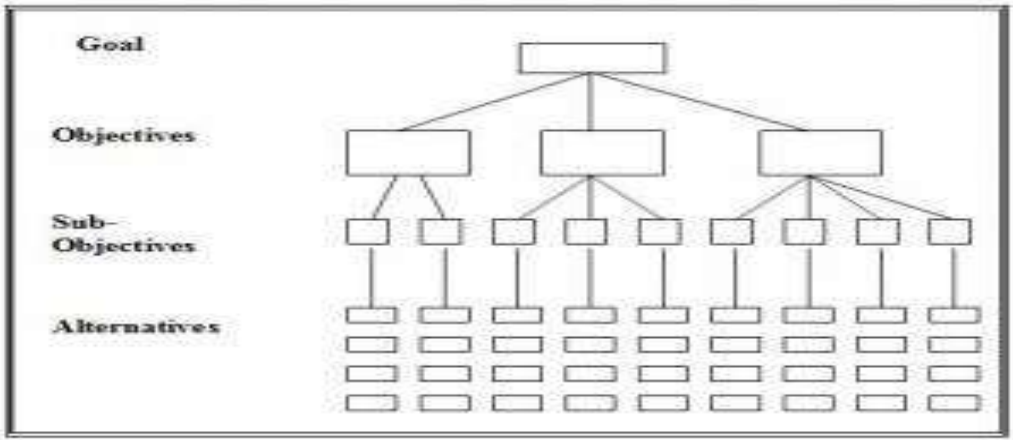
5. Menghitung nilai eigen dan menguji konsistensinya. Jika tidak konsisten maka pengambilan data diulangi.
6. Mengulangi langkah 3,4, dan 5 untuk seluruh tingkat hirarki.
7. Menghitung vektor eigen dari setiap matriks perbandingan berpasangan yang merupakan bobot setiap elemen untuk penentuan prioritas elemen-elemen pada tingkat hirarki terendah sampai mencapai tujuan. Penghitungan dilakukan lewat cara menjumlahkan nilai setiap kolom dari matriks, membagi setiap nilai dari kolom dengan total kolom yang bersangkutan untuk memperoleh normalisasi matriks, dan menjumlahkan nilai-nilai dari setiap baris dan membaginya dengan jumlah elemen untuk mendapatkan rata-rata.
8. Memeriksa konsistensi hirarki. Adapun yang diukur dalam *Analytical Hierarchy Process* adalah rasio konsistensi dengan melihat *index* konsistensi. Konsistensi yang diharapkan adalah yang mendekati sempurna agar menghasilkan keputusan yang mendekati valid. Walaupun sulit untuk mencapai yang sempurna, rasio konsistensi diharapkan kurang dari atau sama dengan 10 %. (*Sylvia Hartati Saragih ; 2013 : 84*)

II.1.4. Prosedur *Analytical Hierarchy Process* (AHP)

Pada dasarnya langkah-langkah dalam metode AHP meliputi :

1. Menyusun hirarki dari permasalahan yang dihadapi.

   Persoalan yang akan diselesaikan, diuraikan menjadi unsur-unsurnya, yaitu kriteria dan alternatif, kemudian disusun menjadi struktur hierarki seperti gambar di bawah ini :



Gambar II.1. Struktur hierarki AHP
Sumber : (Yuni Sugiarti ; 2013 ; 84)

2. Penilaian kriteria dan alternatif

   Kriteria dan alternatif dinilai melalui perbandingan berpasangan. Menurut Saaty (1988), untuk berbagai persoalan, skala 1 sampai 9 adalah skala terbaik dalam mengekspresikan pendapat. Nilai dan definisi pendapat kualitatif dari skala perbandingan Saaty dapat dilihat pada tabel 1.

Tabel 1 : Skala Penilaian Perbandingan Berpasangan

Intensitas	Kepentingan Keterangan
1	Kedua elemen sama pentingnya
3	Elemen yang satu sedikit lebih penting daripada elemen yang lainnya
5	Elemen yang satu lebih penting daripada yang lainnya
7	Satu elemen jelas lebih mutlak penting daripada elemen lainnya
9	Satu elemen mutlak penting daripada elemen lainnya
2,4,6,8	Nilai-nilai antara dua nilai pertimbangan-pertimbangan yang berdekatan

II.2. Sistem Informasi

Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisai dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan. Namun, secara sederhana sistem informasi dapat didefinisikan sebagai suatu sistem yang bertujuan untuk menyimpan, memproses dan mengomunikasikan informasi. (*Purnomo Budi Santoso ; 2013 ; 61*)

Sistem informasi adalah kombinasi antara prosedur kerja, informasi, orang dan teknologi informasi yang diorganisasikan untuk mencapai tujuan dalam sebuah organisasi. Sistem informasi dapat didefinisikan sebagai sekumpulan komponen yang saling berhubungan, mengumpulkan, memproses, menyimpan dan mendistribusikan informasi untuk menunjang pengambilan keputusan dan pengawasan dalam suatu organisasi. *(Dera Kristia Anggraini ; 2013 ; 61*)

Adapun kesimpulan dari Sistem informasi adalah sebagai berikut :

1. Sistem informasi dapat didefinisikan sebagai suatu sistem yang bertujuan untuk menyimpan, memproses dan mengomunikasikan informasi.
2. Sistem diklasifikasi sebagai sistem abstrak dan sistem fisik. Sistem abstrak merupakan sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik. Sedangkan sistem fisik merupakan sistem yang ada secara fisik.

II.3. *Database*

Database merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan diperangkat keras komputer dan digunakan diperangkat lunak untuk memanipulasinya. *Database* merupakan salah satu komponen yang sangat penting dalam sistem informasi, karena merupakan basis sistem dalam menyediakan informasi bagi para pemakai. (*Indra Warman ; 2012 ; 45*)

Basis Data adalah sekumpulan tabel-tabel yang saling berelasi, relasi tersebut bisa ditunjukkan dengan kunci dari tabel yang ada. Basis data juga bisa didefinisikan sebagai suatu kumpulan dari data yang tersimpan dan diatur atau diorganisasikan sehingga data tersebut bisa diambil atau dicari dengan mudah dan efisien, kumpulan data (elementer) yang secara logik berkaitan dalam merepresentasikan fenomena atau fakta secara terstruktur dalam domain tertentu untuk mendukung aplikasi pada sistem tertentu. Basis data adalah kumpulan data yang saling terkait digunakan untuk memenuhi kebutuhan tertentu.

Basis data ialah suatu kumpulan data yang disusun dalam bentuk tabel-tabel yang saling berkaitan maupun berdiri sendiri dan disimpan secara bersama-

sama pada suatu media. Basis data dapat digunakan oleh satu atau lebih program aplikasi secara optimal, data disimpan tanpa mengalami ketergantungan pada program yang akan menggunakannya. (*Anis nurhanafi ; 2013 ; 3*)

Adapun kesimpulan dari *Database* adalah sebagai berikut :

1. Basis data terdiri dari sejumlah tabel-tabel yang saling berelasi dan dapat diatur serta diorganisasikan sehingga data tersebut dapat dicari secara efisien.
2. *Database* merupakan salah satu komponen yang sangat penting dalam sistem informasi, karena merupakan basis sistem dalam menyediakan informasi bagi para pemakai.

II.4. *MySql*

MySQL Server 2000 adalah suatu Perangkat lunak *Relational Database Mangement system* (RDBMS) yang handal. Didesain untuk mendukung proses transaksi yang besar (seperti order entri yang *online*, inventori, akuntansi atau manufaktur). *MySQL Server* akan secara otomatis menginstal enam *database* utama, yaitu master, model, *tempdb*, *pubs*, *Northwind*, dan *Msdb*. (*Anis nurhanafi ; 2013 ; 5*)

MySQL (My Structure Query Language) adalah salah satu DataBase Management System (DBMS). MySQL berfungsi untuk mengelola database menggunakan bahasa SQL. MySQL bersifat open source sehingga kita bisa menggunakannya secara gratis. Pemrograman PHP juga sangat mendukung/support dengan database MySQL. (*Harun Al-Rosyid; 2013 ; 2*)

Adapun kesimpulan dari MySQL (*My Structure Query Language*) adalah sebagai berikut :

1. MySql database Managemen Sytem mengelola database menggunakan bahasa SQL dengan pemograman PHP juga mendukung proses desain-desain yang mendukung.
2. MySql digunakan sebagai pengelola database dan bersifat open source.

II.5. *Visual Basic*

Visual Basic adalah salah satu bahasa pemrograman komputer. Bahasa pemrograman adalah perintah-perintah yang dimengerti oleh komputer untuk melakukan tugas-tugas tertentu. Bahasa pemrograman *Visual Basic* , yang dikembangkan oleh *Microsoft* sejak tahun 1991, merupakan pengembangan dari pendahulunya yaitu bahasa pemrograman BASIC (*Beginner's All-purpose Symbolic Instruction Code*) yang dikembangkan pada era 1950-an. *Visual Basic* merupakan salah satu *Development Tool* yaitu alat bantu untuk membuat berbagai macam program komputer, khususnya yang menggunakan sistem operasi *Windows*. Menurut Widodo (2004 : 86) "*Visual Basic* ialah bahasa pemrograman *event-driven* yang berasal dari BASIC. *Event driven* artinya program menunggu sampai adanya respon dari pemakai berupa kejadian tertentu, misalnya tombol diklik, atau menu dipilih". (*Anis nurhanafi, Sukadi, 2013 ; 2-3*)

Visual basic adalah bahasa pemrograman yang digunakan untuk membuat aplikasi *windows* yang berbasis grafis *(GUI = Grafic User Interface)*. Visual Basic merupakan *Event Driven Programming* (Program terkendali kejadian) artinya program menunggu sampai adanya respon dari pemakai berupa event atau kejadian tertentu, misalnya : memilih menu utama, file, dsb. Ketika event terdeteksi, kode yang berhubungan dengan event akan dijalankan (Suryo A,

2002). Microsoft Visual Basic 6.0 adalah bahasa pemrograman yang bekerja dalam lingkup windows. Ms.Visual Basic 6.0 dapat memanfaatkan kemampuan ms.windows secara optimal. Kemampuannya dapat dipakai untuk merancang aplikasi yang berpenampilan seperti program aplikasi lainnya berbasis ms.windows. Struktur aplikasi visual basic terdiri dari *form*, kontrol, *properties, method, procedure, dan module. Form* adalah windows atau jendela dimana merupakan tempat untuk membuat *user interface* atau tampilan, kontrol adalah tampilan berbasis grafis yang dimasukkan pada form untuk membuat interaksi dengan pemakai *(textbox, label, scroll bar), properties* adalah nilai atau karakteristik yang dimiliki oleh sebuah obyek Visual Basic, contoh : *name, caption, size, dll*, *method* adalah serangkaian perintah yang sudah tersedia pada suatu obyek yang dapat diminta untuk mengerjakan tugas khusus, *procedure* kejadian adalah kode yang berhubungan dengan suatu obyek, kode ini akan dieksekusi ketika ada respon dari pemakai berupa event tertentu, dan *module* adalah kumpulan dari prosedur umum, deklarasi variabel dan definisi konstanta yang digunakan aplikasi. (*Tominanto ; 2010 : 9*)

Adapun kesimpulan dari *Visual Basic* adalah sebagai berikut :

1. *Visual Basic* merupakan Program pengendali kejadian artinya program menunggu sampai adanya respon dari pemakai berupa event atau kejadian tertentu.
2. *Visual Basic* merupakan salah satu *Development Tool* yaitu alat bantu untuk membuat berbagai macam program komputer, khususnya yang menggunakan sistem operasi *Windows*.

II.6. UML *(Unified Modelling Language)*

Unified Modelling Language (UML) adalah sebuah bahasa yang sudah menjadi standar dalam industri untuk merancang, menspesifikasi dan mendokumentasi sistem perangkat lunak. *UML* memberikan standar penulisan tersendiri pada sebuah sistem *blue print*, yang mencakup konsep proses bisnis, penulisan kelas-kelas pada bahasa program yang spesifik, skema database dan komponen-komponen yang dibutuhkan dalam sistem piranti lunak (*Arzan Muharom* ; 2013 : 2)

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan

piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C. Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*). Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: *metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock*, dsb. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan group/perusahaan lain yang menggunakan metodologi yang berlainan. Dimulai pada bulan Oktober 1994 *Booch, Rumbaugh dan Jacobson*, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease *draft* pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh *Object Management Group* (OMG – http://www.omg.org).

Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. *Booch, Rumbaugh* dan *Jacobson* menyusun tiga buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek. *(Yuni Sugiarti ; 2013 : 33)*

Adapun kesimpulan dari *Unified Modelling Language* (UML) adalah sebagai berikut :

1. *Unified Modelling Language* (UML) merupakan pemodelan dan Skema database serta komponen-komponen yang dibutuhkan dalam sistem piranti lunak.
2. *Unified Modelling Language* (UML) memiliki notasi, Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak.

Dalam pembuatan skripsi ini penulis menggunakan diagram *Use Case* yang terdapat di dalam UML. Adapun maksud dari *Use Case* Diagram diterapkan dibawah ini.

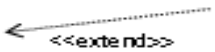
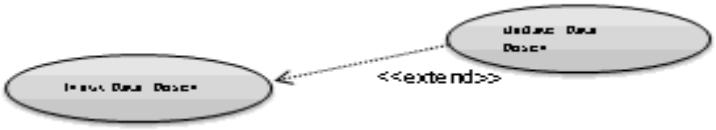
1. *Use Case Diagram*

Use Case Diagram yang mana peroses yang dilakukan dimulai dari mengidentifikasi aktor dan use case dengan merancang aplikasi yang akan dikembangkan, menggambarkan aliran control untuk mengetahui hubungan aktor dan objek, menggambarkan komunikasi antar objek dan aktor, menggambarkan perubahan keadaan suatu objek pada aplikasi kelas tertentu, memodelkan prilaku use case serta objek pada aplikasi dan menggambarkan perubahan suatu objek

pada kelas tertentu. Masukan pada tahapan ini yaitu informasi aplikasi pada penelitian sebelumnya, data – data hasil dari tahapan survey dan metode yang akan digunakan pada tahap user design. Sedangkan hasil atau Output dari tahapan ini berupa informasi actor dan use case yang terlibat, informasi interaksi antar kelas, actor dan objek, informasi fitur apa saja yang akan diterapkan pada pengembangan aplikasi sunda selanjutnya serta desain atau gambaran dari aplikasi tersebut. Untuk sumber daya yang digunakan adalah laptop denagn koneksi internet dan Argo UML. *(Arzan Muharom; 2013 : 3)*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem. Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use*

case juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain. *(Yuni Sugiarti ; 2013 : 41)*

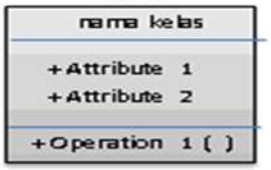
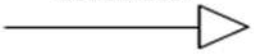
Simbol	Deskripsi
Use Case 	fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit dan aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama use case
Aktor  nama aktor	orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
Asosiasi / association	komunikasi antara aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor
Extend  <<extend>>	relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan dapat berdiri sendiri walaupun tanpa use case tambahan itu; mirip dengan prinsip inheritance pada pemrograman berorientasi objek; biasanya use case tambahan memiliki nama depan yang sama dengan use case yang ditambahkan, arah panah menunjukan pada use case yang dituju contoh : 
Include <<include>> 	relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya atau sebagai syarat dijalankan use case ini. Ada dua sudut pandang yang cukup besar mengenai include di use case, include berarti use case yang ditambahkan akan selalu dipanggil saat use case tambahan dijalankan, contoh : 

Gambar II.2. *Use Case* Diagram
Sumber : (Yuni Sugiarti ; 2013 ; 42)

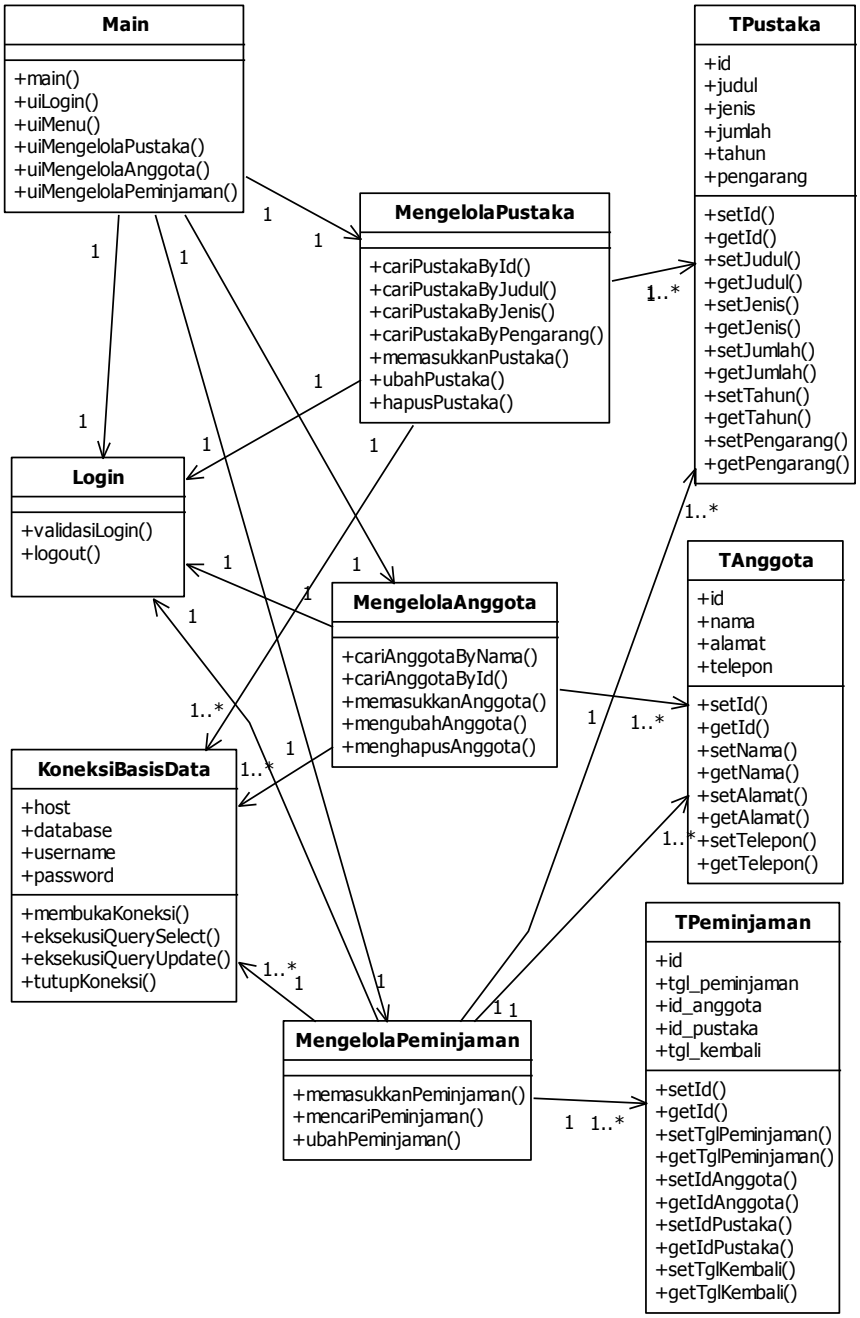
2. *Class Diagram*

Class Diagram yang terdapat pada aplikasi ini terdiri dari super class yaitu Activity yang akan menjadi menu utama pada aplikasi, sedangkan class untuk pembahasan yaitu class yang memiliki subclass yang berfungsi sebagai alat bantu untuk mendapat informasi dari dinas pariwisata setempat. Activity merupakan rancangan aplikasi untuk menu utama yang mana activity tersebut merujuk pada skenario Use case dan aktivitas pemodelan Activity Diagram, sedangkan untuk merujuk pada Use Case Diagram yang mana masing – masing tersebut merupakan fitur yang da pada aplikasi. Untuk merujuk pada skenario Use Case dan pemodelan yang telah dilakukan. *(Arzan Muharom; 2013 : 5)*

Diagram kelas atau *class* diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Berikut adalah simbol-simbol pada diagram kelas :

Simbol	Deskripsi
Package 	Package merupakan sebuah bungkusan dari satu atau lebih kelas
Operasi 	Kelas pada struktur sistem
Antarmuka / interface 	sama dengan konsep interface dalam pemrograman berorientasi objek
Asosiasi 	relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan multiplicity
Asosiasi berarah/directed asosiasi	relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan multiplicity
Generalisasi 	relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus)
Kebergantungan / defedency	relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi 	relasi antar kelas dengan makna semua-bagian (whole-part)

Gambar II.3. *Class* Diagram
Sumber : (Yuni Sugiarti ; 2013 : 59)



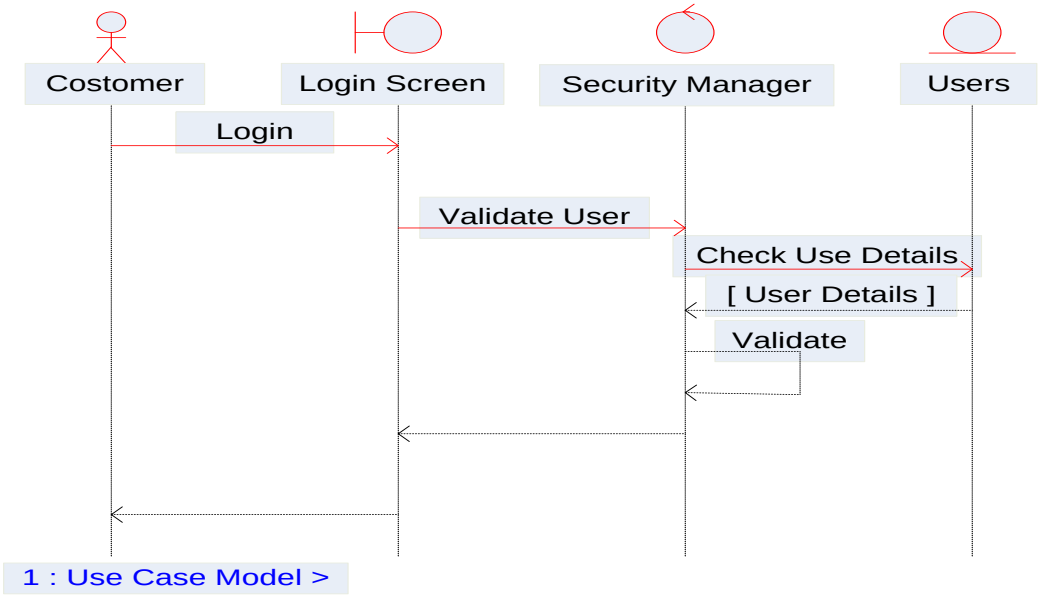
Gambar II.4. Contoh Class Diagram

Sumber : (Yuni Sugiarti ; 2013 : 63)

3. *Sequence Diagram*

Diagram *Sequence* menggambarkan kelakuan/prilaku objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram *sequence* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya diagram *sequence* yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram *sequence* sehingga semakin banyak *use case* yang didefinisikan maka diagram *sequence* yang harus dibuat juga semakin banyak.



Gambar II.5. Contoh *Sequence* Diagram
Sumber : (Yuni Sugiarti ; 2013 : 63)

4. *Activity Diagram*

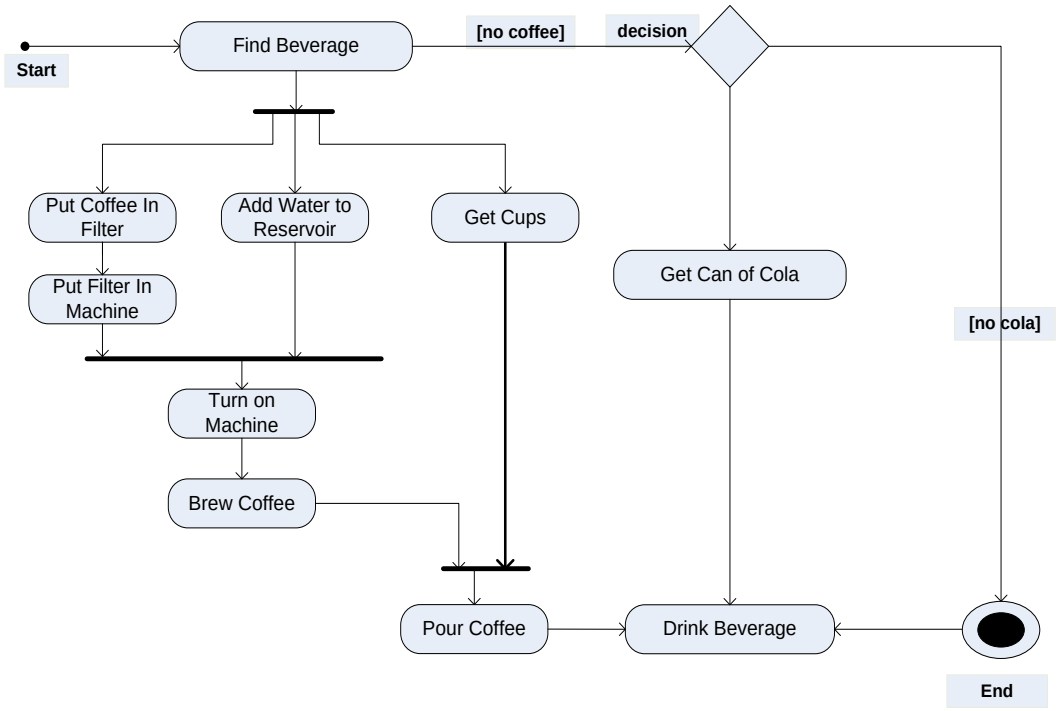
Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal.

Activity diagram dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.



Gambar II.6. Activity Diagram
Sumber : (Yuni Sugiarti ; 2013 : 76)