BAB II

TINJAUAN PUSTAKA

II.1. Sistem Pendukung Keputusan

Sistem pendukung keputusan (SPK) atau *Decision Support Systems (DSS)* adalah sistem informasi interaktif yang menyediakan informasi, pemodelan, dan pemanipulasian data yang digunakan untuk membantu pengambilan keputusan pada situasi yang semiterstruktur dan situasi yang tidak terstruktur di mana tidak seorang pun tahu secara pasti bagaimana keputusan seharusnya dibuat. Konsep *DSS* dikemukakan pertama kali oleh Scott-Morton pada tahun 1971. Beliau mendefenisikan cikal bakal *DSS* tersebut sebagai ”Sistem berbasis komputer interaktif, yang membantu pengambil keputusan menggunakan data dan model untuk memecahkan persoalan-persoalan tidak terstruktur”.

DSS dibuat sebagai reaksi atas ketidakpuasan terhadap TPS dan MIS. Sebagaimana diketahui, TPS lebih memfokuskan diri dari pada perekaman dan pengendalian transaksi yang merupakan kegiatan yang bersifat berulang dan terdefenisi dengan baik, sedangkan MIS lebih berorientasi pada penyediaan laporan bagi manajemen yang sifatnya tidak fleksibel. *DSS* lebih ditujukan untuk mendukung manajemen dalam melakukan pekerjaan yang bersifat analitis, dalam situasi yang kurang terstruktur dan dengan kriteria yang kurang jelas. *DSS* tidak dimaksudkan untuk mengotomasikan pengambilan keputusan, tetapi memberikan perangkat interaktif yang memungkinkan pengambil keputusan dapat melakukan

berbagai analisis dengan menggunakan model-model yang tersedia. (Abdul Kadir, 2014, Hal : 108).

Adapun beberapa karakteristik yang terdapat pada sistem pendukung keputusan adalah sebagai berikut :

1. Menawarkan keluasan, kemudahan beradaptasi, dan tanggapan yang cepat.
2. Memungkinkan pemakai memulai dan mengendalikan masukan dan keluaran.
3. Dapat dioperasikan dengan sedikit atau tanpa bantuan pemrogram professional.
4. Menyediakan dukungan untuk keputusan dan permasalahan yang solusinya tak dapat ditentukan di depan.
5. Menggunakan analisis data dan perangkat pemodelan yang canggih. (Abdul Kadir, 2014, Hal : 108).

II.2. Metode *Profile Matching*

Metode *profile matching* atau pencocokan profil adalah metode yang sering digunakan sebagai mekanisme dalam pengambilan keputusan dengan mengasumsikan bahwa terdapat tingkat *variable predictor* yang ideal yang harus dipenuhi oleh subyek yang diteliti, bukannya tingkat minimal yang harus dipenuhi atau dilewati. Dalam proses *profile matching* secara garis besar merupakan proses membandingkan antara setiap kriteria setiap penilaian dalam sebuah proposal usulan penelitian yang diajukan sehingga diketahui perbedaan skornya (disebut juga *gap*), semakin kecil *gap* yang dihasilkan maka bobot nilainya semakin besar yang berarti memiliki peluang lebih besar untuk prioritas kelayakan/kelulusan.

(Edi Faizal, 2014, Hal : 61). Nilai gap dapat dihitung menggunakan persamaan (1). Sedangkan pembobotan nilai gap ditentukan berdasarkan Tabel II.7.

***GAP* = *Profile* proposal – *Profile* ideal** ……...(1)

Langkah selanjutnya adalah menghitung nilai *core factor* dan *secondary factor*. *Core factor* merupakan kriteria penilaian yang paling utama harus terkandung dalam sebuah proposal penelitian. Perhitungan *core factor* menggunakan persamaan (2).

Tabel II.1. Pembobotan Nilai Gap

No	Selisih	Bobot	Keterangan
1	0	5	Tidak ada selisih skor kriteria
2	1	4,5	Kriteria kelebihan 1 level
3	-1	4	Kriteria kekurangan 1 level
4	2	3,5	Kriteria kelebihan 2 level
5	-2	3	Kriteria kekurangan 2 level
6	3	2,5	Kriteria kelebihan 3 level
7	-3	2	Kriteria kekurangan 3 level

(Sumber : Edi Faizal, 2014, Hal : 63)

$$NCF = \frac{\sum NC(kriteria)}{\sum IC}$$ ……(2)

Keterangan:

NCT : Nilai rata–rata *core factor*

NC : Jumlah total nilai *core factor*

IC : Jumlah item *core factor*

Sedangkan *secondary factor* merupakan item-item selain yang ada pada faktor utama (*core factor*). *Secondary factor* dihitung menggunakan persamaan (3).

$$NSF = \frac{\sum NS(kriteria)}{\sum IS}$$ ……..(3)

Keterangan:

NST : Nilai rata–rata *secondary factor*

NS : Jumlah total nilai *secondary factor*

IS : Jumlah item *secondary factor*

Selanjutnya perhitungan nilai total berdasar nilai dari *core* dan *secondary factor* yang digunakan sebagai kriteria penilaian yang berpengaruh terhadap kelulusan proposal penelitian. Perhitungan dapat dilakukan menggunakan persamaan (4).

$$N\ (Tot\_kriteria) = (x)\%NCF + (x)\%NSF$$ ……..(4)

Keterangan:

NCT : Nilai rata–rata *core factor*

NST : Nilai rata–rata *secondary factor*

NT : Nilai total kriteria penilaian

Langkah terakhir adalah perhitungan ranking, yang dilakukan dengan menggunakan persamaan (5).

$$Ranking = (x)\%N1 + (x)\%N2 + (x)\%Nn$$ ……(5)

Keterangan:

N1, N2, Nn : Nilai total per kriteria

(x)% : Persentase nilai kriteria

II.3. Basis Data Dan *DBMS*

Basis data dapat didefenisikan sebagai koleksi dari data-data yang terorganisasi sedemikian rupa sehingga data mudah disimpan dan dimanipulasi (diperbarui, dicari, diolah dengan perhitungan-perhitungan tertentu, serta dihapus). Secara teoritis, basis data tidak harus berurusan dengan komputer (misalnya, catatan belanja hari ini yang dibuat oleh seorang ibu rumah tangga juga merupakan basis data dalam bentuk yang sangat sederhana). (Adi Nugroho, 2011, Hal : 4).

Menurut Abdul Kadir (2014) basis data (*database)* adalah suatu pengorganisasian sekumpulan data yang saling terkait sehingga memudahkan aktifitas untuk memperoleh informasi. Basis data dimaksudkan untuk mengatasi problem pada sistem yang memakai pendekatan berbasis berkas.

Untuk mengelola basis data diperlukan perangkat lunak yang disebut *Database Management System* (*DBMS*). *DBMS* adalah perangkat lunak sistem yang memungkinkan para pemakai membuat, memelihara, mengontrol dan mengakses basis data dengan cara yang praktis dan efisien. *DBMS* dapat digunakan untuk mengakomodasikan berbagai macam pemakai yang memiliki kebutuhan akses yang berbeda-beda. (Abdul Kadir, 2014, Hal : 218).

Umumnya *DBMS* menyediakan fitur-fitur sebagai berikut :

- Independensi data-program

  Karena basis data ditangani oleh *DBMS*, program dapat dipilih sehingga tidak tergantung pada struktur data dalam basis data. Dengan perkataan lain, program tidak akan terpenaruh sekiranya bentuk fisik data diubah.

- Keamanan

  Keamanan dimaksudkan untuk mencegah pengaksesan data oleh orang yang tidak berwewenang.

- Integritas

  Hal ini ditujukan untuk menjaga agar data selalu dalam keadaan yang valid dan konsisten.

- Konkurensi

  Konkurensi memungkinkan data dapat diakses oleh banyak pemakai tanpa menimbulkan masalah.

- Pemulihan (*recovery*)

  DBMS menyediakan mekanisme untuk mengembalikan basis data ke keadaan semula yang konsisten sekiranya terjadi gangguan perangkat keras atau kegagalan perangkat lunak.

- Katalog Sistem

  Katalog Sistem adalah deskripsi tentang data yang terkandung dalam basis data yang dapat diakses oleh pemakai.

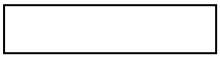
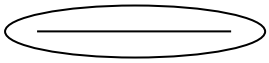
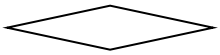
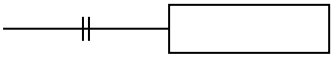
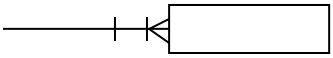
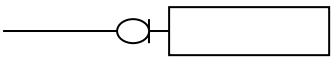
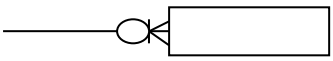
- Perangkat Produktivitas

  Untuk menyediakan kemudahan bagi pemakai dan meningkatkan produktivitas, *DBMS* menyediakan sejumlah perangkat produktivitas seperti pembangkit *query* dan pembangkit laporan. (Abdul Kadir, 2014, Hal : 219).

II.4. *Entity Relationship Diagram*

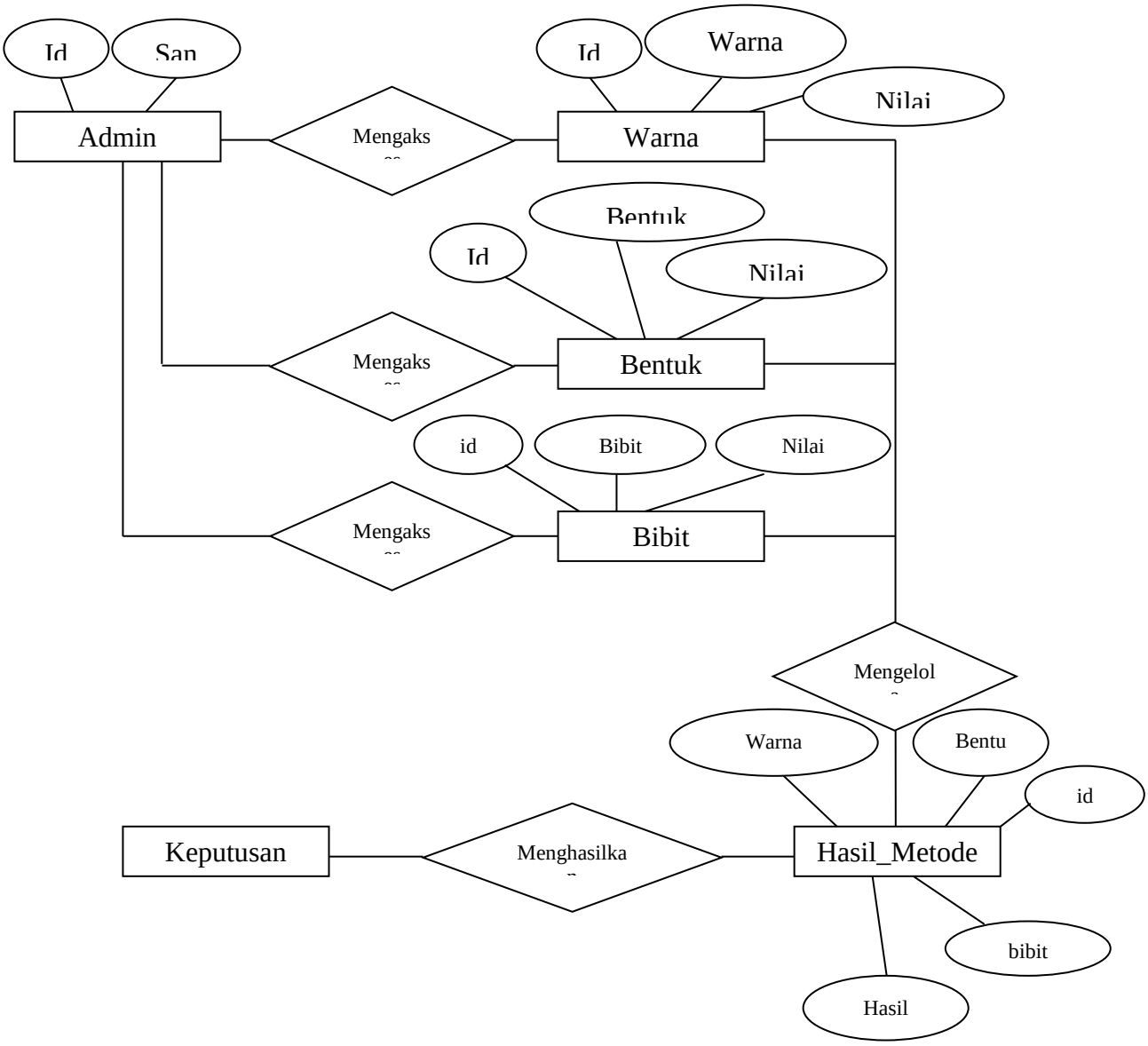
Entity Relationship Diagram (ERD) adalah bagian yang menunjukkan hubungan antara entity yang ada dalam sistem. Simbol-simbol yang digunakan dapat dilihat dari tabel II.6. (Yuhendra, M.T, Dr. Eng dan Riza Eko Yulianto, 2015, Hal : 70).

Tabel II.2. Simbol Yang Digunakan Pada *Entity Relationship Diagram (ERD)*

SIMBOL	KETERANGAN
	*Entity*
	Atribut Dan *Entity*
	Atribut Dan *Entity* Dengan *Key* (Kunci)
	Relasi Atau Aktifitas Antar *Entity*
	Hubungan Satu Dan Pasti
	Hubungan Banyak Dan Pasti
	Hubungan Satu Tapi Tidak Pasti
	Hubungan Banyak Tapi Tidak Pasti

(Sumber : Yuhendra dan Riza Eko Yulianto; 2015, Hal : 70)

Berikut adalah contoh penggunaan *Entity Relationship Diagram* (*ERD*) :



Gambar II.1. Contoh *Entity Realitionship Diagram* (*ERD*)

II.5. Kamus Data

Kamus data merupakan sebuah daftar yang terorganisasi dari elemen data yang berhubungan dengan sistem, dengan definisi yang tegar dan teliti sehingga pemakai dan analis sistem akan memiliki pemahaman yang umum mengenai *input*, *output*, komponen penyimpanan. (Yusi Ardi Binarso, 2012, Hal : 74).

Contoh kamus data untuk entitas wisuda dan alumni yang digunakan berdasarkan sistem informasi alumni teknik informatika adalah sebagai berikut :

1. Data Wisuda

   Wisuda = @idWisuda + bulan + tahun + jumlahPeserta

   @Wisuda = { integer }

   Bulan = [januari|…|Desember]

   Tahun = [2000|…|2999]

   Jumlah Peserta = { integer }

   Integer = [0-9]

2. Data Alumni

   Alumni = @NIM + namaLengkap + password + email + tglLahir + jenisKelamin + noTelp + alamatAsal + kotaAsal + alamatSekarang + kotaSekarang + instansi + jabatan + judulTA + lamaTA + tglLulus + lamaStudi + IPK + foto + jawabanPK + kunciAktivasi + statusAktif + idWisuda + idPertanyaan

   @NIM = 1 { character } 14

   namaLengkap = 1 { character } 50

   password = 1 { character } 50

   email = 1 { character } 25

   tglLahir = date

   jenisKelamin = { L | P }

   noTelp = 1 { character } 20

   alamatAsal = 1 { character } 100

kotaAsal = 1 { character } 20

alamatSekarang = 1 { character } 100

kotaSekarang = 1 { character } 20

instansi = 1 { character } 50

jabatan = 1 { character } 50

judulTA = 1 { character } 250

lamaTA = { integer }

tglLulus = date

lamaStudi = { integer }

IPK = decimal

Foto = 1 { character } 50

JawabanPK = 1 { character } 20

kunciAktivasi = 1 { character } 65

statusAktif = [aktif | non aktif]

idWisuda = *dapat dilihat pada data Wisuda

idPertanyaan = *dapat dilihat pada data Pertanyaan_Keamanan

character = [A-Z | a-z | 0-9 | |]

integer = [0-9]

II.6. Normalisasi

Normalisasi dapat dipahami sebagai tahapan-tahapan yang masing-masing berhubungan dengan bentuk normal. Bentuk normal adalah keadaan relasi yang dihasilkan dengan menerapkan aturan sederhana berkaitan dengan konsep

kebergantungan fungsional pada relasi yang bersangkutan. (Adi Nugroho, 2011, Hal : 199). Kita akan menggambarkannya secara garis besar sebagai berikut :

1. Bentuk Normal Pertama (1NF/ *First Normal Form*)

   Bentuk normal pertama adalah suatu bentuk relasi dimana atribut bernilai banyak (*multivalues attribute*) telah dihilangkan sehingga kita akan menjumpai nilai tunggal (mungkin saja nilai *null*) pada perpotongan setiap baris dan kolom.

2. Bentuk Normal Kedua (2ND/ *Second Normal Form*)

   Semua kebergantungan fungsional yang bersifat sebagian (*particial functional dependency*) telah dihilangkan.

3. Bentuk Normal Ketiga (3RD/ *Thrid Normal Form*)

   Semua kebergantungan transitif (*transitive dependency*) telah dihilangkan.

4. Bentuk Normal *Boyce-Codd* (BCNF/ *Boyce-Codd Normal Form*)

   Semua anomaly yang tersisa dari hasil penyempurnaan kebergantungan fungsional sebelumnya telah dihilangkan.

5. Bentuk Normal Keempat (4NF/ *Fourth Normal Form*)

   Semua kebergantungan bernilai banyak telah dihilangkan.

6. Bentuk Normal Kelima (5NF/ *Fifth Normal Form*)

   Semua anomaly yang tertinggi telah dihilangkan. (Adi Nugroho, 2011, Hal : 200).

Berikut ini adalah contoh normalisasi :

Bentuk tidak normal

Tabel II.3. Bentuk Tidak Normal

ID	Admin	Kriteria	Bobot
1	Nisa	Warna	3
2	Yono	Bibit	2

Bentuk normal pertama

Tabel II.4. Bentuk Normal Pertama

ID	Kriteria	Bobot
1	Warna	3
2	Bibit	2

Bentuk normal kedua

Tabel II.5. Bentuk Normal Kedua

Kriteria	Bobot
Warna	3
Bibit	2

II.7. *SQL Server* 2008

SQL Server 2008 adalah sebuah *RDBMS (Relational Database Management System)* yang sangat powerful dan telah terbukti kekuatannya dalam mengolah data. Dalam versi terbarunya ini, *SQL Server* 2008 memiliki banyak fitur yang bisa dihandalkan untuk meningkatkan performa *database. SQL Server* 2008 memiliki suatu *GUI (Graphic User Interface)* yang kita gunakan untuk melakukan aktivitas sehari hari berkaitan dengan database,seperti menulis *T-SQL*, melakukan *backup* dan *restore database*, melakukan security database terhadap

aplikasi, dan sebagainya. Pada *GUI* tersebut kita bisa melakukan settingan terhadap *SQL Serve*r untuk berkerja lebih optimal.Settingan juga bisa dilakukan menggunakan script untuk memudahkan developer mengubah Setting Opsions pada *SQL Server* 2008. (Ruslan, 2013, Hal : 39).

Microsoft merilis *SQL Server* 2008 dalam beberapa versi yang disesuaikan dengan segment-segment pasar yang dituju.Versi-versi tersebut adalah sebagai berikut. Menurut cara pemrosesan data pada prosesor maka *Microsoft* mengelompokkan produk ini berdasarkan 2 jenis yaitu :

1. Versi 32-bit(x86), yang biasanya digunakan untuk komputer dengan *single processor* (*Pentium* 4) atau lebih tepatnya prosesor 32 bit dan sistem operasi *Windows XP*.
2. Versi 64-bit(x64), yang biasanya digunakan untuk komputer dengan lebih dari satu prosesor (Misalnya *Core* 2 *Duo*) dan sistem operasi 64 bit seperti *Windows XP* 64, *Vista, dan Windows* 7.

Sedangkan secara keseluruhan terdapat versi-versi seperti berikut ini :

1. Versi *Compact*, ini adalah versi "Tipis" dari semua versi yang ada. Versi ini seperti versi *desktop* pada *SQL Server* 2000. Versi ini juga digunakan pada *handled device* seperti *Pocket PC, PDA, SmartPhone, Tablet PC*.
2. Versi *Express*, ini adalah versi "Ringan" dari semua versi yang ada(tetapi versi ini berbeda dengan versi *compact*) dan paling cocok untuk latihan para pengembang aplikasi. Versi ini memuat *Express Manager Standar*, integrasi dengan *CLR* dan *XML*. (Wenny Widya, dkk, 2011, Hal : 3).

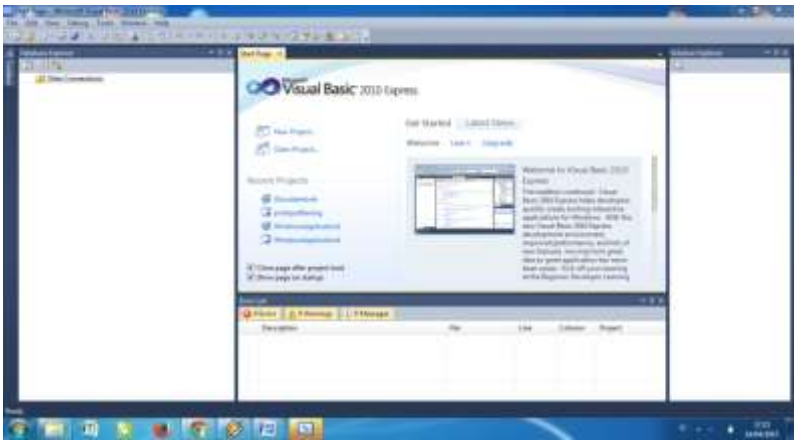
II.8. *Microsoft Visual Basic 2010*

Visual Basic 2010 merupakan salah satu bagian dari produk pemrograman terbaru yang dikeluarkan oleh *Microsoft*, yaitu *Microsoft Visual Studio 2010. Visual Studio* merupakan produk pemrograman andalan dari *mocrosoft corporation*, dimana di dalamnya berisi beberapa jenis *IDE* pemrograman seperti *Visual Basic, Visual C++, Visual Web Developer, Visual C#,* dan *Visual F#*.

Semua *IDE* pemrograman tersebut sudah mendukung penuh implementasi *.Net Framework* terbaru, yaitu *.Net Framework 4.0* yang merupakan pengembangan dari *.Net Framework 3.5*. Adapun database standar yang disertakan adalah *Microfot SQL Server 2008 express*.

Visual Basic 2010 merupakan versi perbaikan dan pengembangan dari versi pendahulunya yaitu *visual basic 2008.* Beberapa pengembangan yang terdapat di dalamnya antara lain dukungan terhadap *library* terbaru dari *Microsoft*, yaitu *.Net Framework 4.0,* dukungan terhadap pengembangan aplikasi menggunakan *Microsoft SilverLight*, dukungan terhaddap aplikasi berbasis *cloud computing*, serta perluasan dukungan terhadap *database-database*, baik *standalone* maupun *database server.* (Wahana Komputer, 2011, Hal : 2).

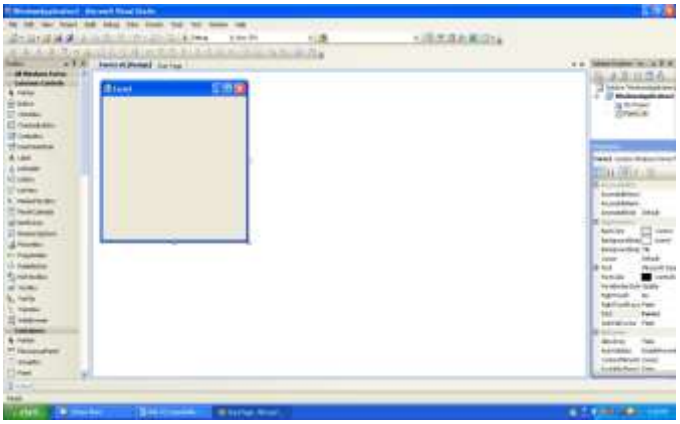
Untuk melihat tampilan *visual studio* 2010 dapat dilihat pada gambar II.2. sebagai berikut :



Gambar II.2. Tampilan Utama Visual Studio 2010

Sumber : (Priyanto Hidayatullah ; 2012 : 23)

Di dalam *Visual Studio.NET* menyediakan tampilan *Interface* yang sangat mudah untuk para pengguna merancang dan memodifikasi bentuk atau *Interface* dari program yang akan dibuat, dimana pada tampilan ini difasilitasi dengan *Tool* dan fasilitas pendukung lainnya untuk mempermudah pengerjaannya.



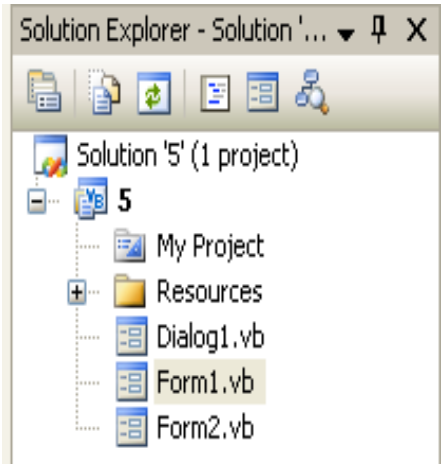
Gambar II.3. Tampilan *Area* Kerja *Visual Studio.NET*

Sumber : (Priyanto Hidayatullah ; 2012 : 23)

Dalam buku Belajar Pemrograman *Visual Basic* terbitan Teknik Informatika Bandung menjelaskan tampilan Microsoft *Visual Studio.NET*. Ada banyak hal penting yang harus diketahui oleh seorang programmer, diantaranya adalah:

a. *Solution Explorer*

Merupakan fitur yang terletak di sebelah kanan atas di bawah menu aplikasi yang digunakan untuk menampilkan daftar desain *form* dengan *struktur tree* dari project yang sedang dibuka. Dengan adanya fitur ini, memudahkan untuk berpindah antardesain *form* yang sudah anda buat secara cepat dan mudah.



Gambar II.4. *Solution Explorer*

Sumber : (Priyanto Hidayatullah ; 2012 : 29)

b. *Properties*

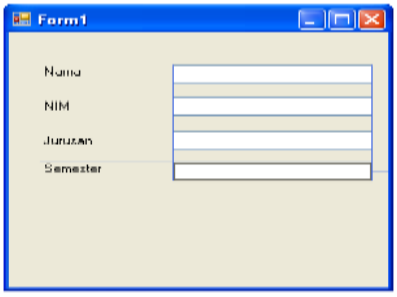
Merupakan fitur yang digunakan untuk melakukan pengaturan properti dari objek-objek yang digunakan dalam desain *form* yang akan dibuat, seperti pengaturan *text, visible, font, name* dan lain sebagainya.

Gambar II.5. Jendela *Properties*

Sumber : (Priyanto Hidayatullah ; 2012 : 30)

c. *Form* Designer

Merupakan fitur yang digunakan untuk membuat desain antarmuka atau *interface* dari aplikasi yang akan dikembangkan. Dengan mengadopsi fitur *click and drop*, proses pemanbahan komponen pada *form* menjadi semakin dinamis dan mudah. Selain itu tersedia pula fitur *guidelines* yang memudahkan anda dalam menata komponen yang terdapat pada desain *form* yang sedang anda kerjakan.

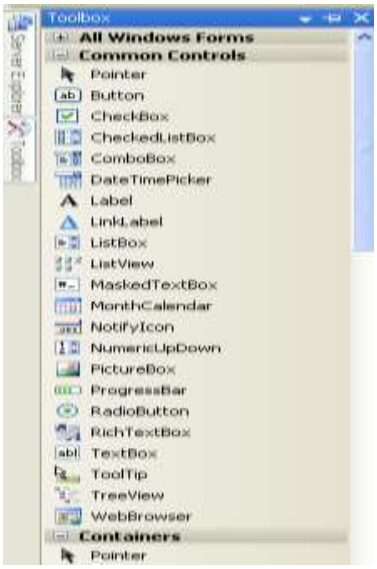


Gambar II.6. *Form Designer*

Sumber : (Priyanto Hidayatullah ; 2012 : 23)

d. *Component Toolbox*

Merupakan fitur *Visual Basic.NET* yang digunakan untuk menampilkan daftar dari komponen baik visual maupun non visual yang bias ditambahkan ke dalam desain *form* yang akan dibuat. Dalam komponen *toolbox*, terdapat berbagai macam kategori sesuai dengan kelompok komponen yang diakses sehingga akan memudahkan dalam mencari komponen yang akan ditambahkan kedalam desain *form*.

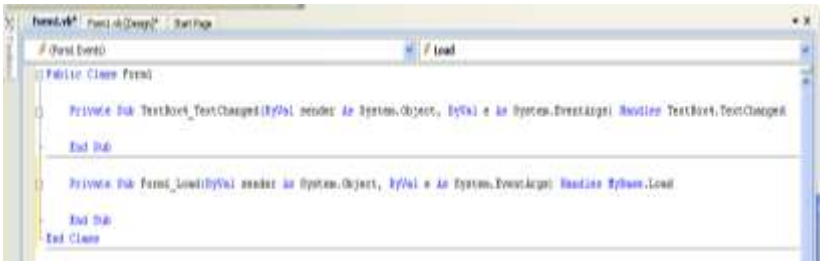


Gambar II.7. *Component Toolbox*

Sumber : (Priyanto Hidayatullah ; 2012 : 28)

e. *Code Editor*

Merupakan fitur yang digunakan untuk menambahkan kode program dari aplikasi atau *project* yang akan dikerjakan.



Gambar II.8. *Code Editor*

Sumber : (Priyanto Hidayatullah ; 2012 : 31)

II.9. *Unified Modeling Language* (*UML*)

Menurut Windu Gata (2013) Hasil pemodelan pada OOAD terdokumentasikan dalam bentuk *Unified Modeling Language* (*UML*). *UML* adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak.

UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. *UML* saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem. (Gellysa Urva dan Helmi Fauzi Siregar, 2015, Hal : 93).

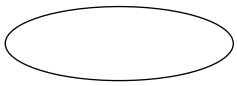
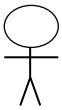
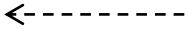
Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan *UML* adalah sebagai berikut:

1. *Use case* Diagram

Use case diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem

informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram dapat dilihat pada tabel II.6 dibawah ini :

Tabel II.6. Simbol *Use Case*

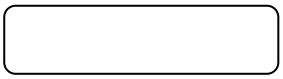
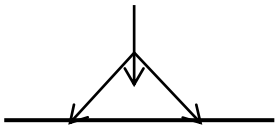
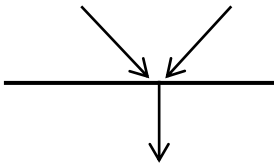
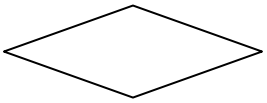
Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *use case*, tetapi tidak memiliki control terhadap *use case*.
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
	Include, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	*Extend,* merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

(Sumber : Gellysa Urva dan Helmi Fauzi Siregar; 2015, Hal : 94)

2. Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram* dapat dilihat pada tabel II.7 dibawah ini:

Tabel II.7. Simbol *Activity Diagram*

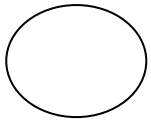
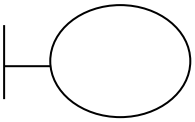
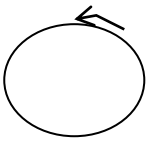
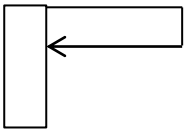
Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true*, *false*.
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

(Sumber : Gellysa Urva dan Helmi Fauzi Siregar; 2015, Hal : 94)

3. Diagram Urutan (*Sequence Diagram*)

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram* dapat dilihat pada tabel II.8 dibawah ini :

Tabel II.8. Simbol *Sequence Diagram*

Gambar	Keterangan
	*Entity Class*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*Boundary Class*, berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan *form* cetak.
	*Control class*, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.
	*Message*, simbol mengirim pesan antar *class*.
	*Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	*Activation*, *activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

(Sumber : Gellysa Urva dan Helmi Fauzi Siregar; 2015, Hal : 95)

4. *Class Diagram* (Diagram Kelas)

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggng jawab entitas yang menentukan perilaku sistem.

Class diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*), *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut.

Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti yang dapat dilihat pada tabel II.9 dibawah ini:

Tabel II.9. *Multiplicity Class Diagram*

Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber : Gellysa Urva dan Helmi Fauzi Siregar; 2015, Hal : 95)