BAB 2
LANDASAN TEORI

2.1 Ringkasan

Ringkasan adalah suatu cara yang efektif untuk menyajikan suatu karangan yang panjang dalam bentuk singkat. Oleh karena itu membuat ringkasan atas sebuah karangan yang panjang dapat diumpamakan sebagai memangkas sebatang pohon sehingga tinggal batang, cabang-cabang dan ranting-ranting yang terpenting beserta daun-daun yang diperlukan, sehingga tampak bahwa esensi pohon masing dipertahankan. Dalam ringkasan keindahan gaya bahasa, ilustrasi, serta penjelasan-penjelasan yang terperinci dihilangkan, sedangkan sari kerangaknya dibiarkan tanap hiasan. Walaupun bentuknya ringkas, namun tetap mempertahankan pikiran pengarang dan pendekatan yang asli [11].

Dalam meringkas sebaiknya menggunakan kalimat tunggal daripada kalimat majemuk. Kalimat majemuk menunjukan bahwa ada dua gagasan atau lebih yang bersifat paralel. Bila memungkinkan ringkaslah kalimat menjadi frasa, selanjutnya frasa menjadi kata. Begitu pula rangkaian gagasan yang penjang hendaknya diganti dengan suatu gagasan sentral saja. Pada umumnya suatu ringkasan ditentukan panjang ringkasan akhirnya, ada yang meringkas seperdua total kata, sepertiga total kata, hingga sepersepuluh total kata. Untuk artikel yang memiliki jumlah kata yang sangat banyak tentu saja hasil ringkasan sepersepuluh menjadi pilihan yang tepat. Sedangkan untuk artikel yang jumlah katanya tidak terlalu banyak maka hasil ringkasan sebanyak seperdua total kata menjadi pilihan yang tepat sehingga hasil ringkasan nantinya masih dapat dibaca dengan struktur hasil ringkasan yang baik.

2.2 Teks

Teks dapat diartikan kumpulan karakter yang membentuk kata yang disusun dan memiliki suatu makna. Teks dapat berupa angka, frasa, kalimat, atau paragraph. Dalam peringkasan teks, teks yang diinginkan dapat berbentuk paragraf yang terdiri dari susunan kalimat yang memiliki informasi yang nantinya

akan diproses untuk digali informasinya akan dimanfaatkan untuk menghasilkan teks yang lebih ringan.

2.3 Abstrak

Abstrak merupakan sebuah ringkasan isi dari sebuah karya tulis ilmiah yang ditujukan untuk membantu seorang pembaca agar dapat dengan mudah dan cepat untuk melihat tujuan dari penulisannya. Di dalam dunia akademik, tulisan pendek ini digunakan oleh institusi/lembaga/organisasi pendidikan sebagai informasi awal atas sebuah penelitian ketika dimasukkan dalam jurnal, konferensi, lokakarya, atau yang sejenisnya. Dalam dunia maya (internet), sebuah abstrak digunakan sebagai gambaran singkat atas sebuah karya tulis ilmiah/penelitian untuk dibaca, sebagaimana halnya sebuah "display" model pakaian dipajang untuk dilihat atau diuji pakai sebelum dibeli. Selanjutnya, bagian lengkap sebuah penelitian dijual kepada mereka yang berminat untuk mendapatkannya.

Untuk membuat abstrak, hal-hal yang perlu diperhatikan adalah sebagai berikut:

1. Struktur paragraf

   Sebuah abstrak ditulis dalam satu paragraf yang menerangkan keseluruhan isi tulisan secara singkat dan jelas. Penulisannya tidak melakukan indentasi pada kalimat pertama paragraf. *Single space* adalah pilihan yang dimiliki oleh penulis untuk menyusun kalimat dalam paragrafnya. Lebih dalam, kadang seorang pembimbing Skripsi/Tesis/Disertasi mengatur hingga pada penggunaan jenis huruf dan ukuran tertentu.

2. Jumlah kata

   Idealnya sebuah paragraf terdiri dari 150 sampai dengan 200 kata. Namun, pertimbangan jumlah kata yang paling tepat dalam penulisan Skripsi, Tesis, ataupun disertasi biasanya bergantung pada pertimbangan pandangan pembimbing (supervisor) yang mendampingi seorang mahasiswa dalam penulisannya. Seorang supervisor harusnya tidak mempertimbangkan jumlah kata sebagai acuan utama penulisan paragraf, karena bagian utama justru isi (content) paragraf.

3. Isi paragraf

Pertama, identifikasi fokus penelitian dijelaskan secara singkat agar pembaca memahami apa yang diamati oleh seorang peneliti di dalam penelitiannya. Kedua, penulis perlu menggambarkan secara jelas desain penelitian yang dilakukan dalam proses pencarian jawaban atau solusi atas persoalan yang diangkat di dalam penelitiannya. Desain langkah penyelesaian masalah ini oleh mahasiswa lazim dikenal dengan istilah Metode Penelitian. Ketiga, selanjutnya penulis akan menjelaskan hasil temuannya kepada pembaca. Beberapa peneliti menganggap hasil temuan yang diungkap tidak perlu mengungkap pembahasan yang dilakukan karena hal itu justru akan membuat pengulangan isi tulisan. Jelas maksudnya karena bagian pembahasan temuan penelitian juga diurai di dalam bagian kesimpulan. Keempat, perlunya bagian kesimpulan di dalam sebuah tulisan juga terlihat di dalam sebuah abstrak yang tetap mendapatkan perhatian penting sebagai bagian akhir dari paragraf. Pada bagian ini kadangkala sejumlah peneliti menyisipkan rekomendasi penelitian namun tanpa pembahasan atau uraian yang panjang. Lebih lanjut, tidaklah lazim sebuah abstrak diisi oleh nama si penulis serta para pembimbing tulisannya, apalagi hal itu ditulis dalam huruf cetak tebal

2.4 Tahapan *Text Mining*

Pada tahap *text mining* ini terdapat lima langkah yaitu:

a. *Case Folding*, yaitu mengubah semua teks ke dalam huruf kecil.
b. *Filtering*, menghapus karakter yang tidak dipakai. Adapun karakter yang akan dihilangkan yaitu : '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', ',', "", '-', '/', '{', '}', '+', '_', '!', '@', '#', '$', '%', '^', '&', '*', '(', ')', '?', '<', '>', '[', ']', '|', '~', '`', ';', ':', '=' dan “’s” untuk dokumen berbahasa Inggris akan dihapus kecuali tanda titik.
c. *Tokenizing*, yaitu memecah kalimat-kalimat yang ada kedalam kata.

d. *Synonim Checking,* yaitu mengganti kata yang ada yang memiliki arti yang sama. Penggantian kata yang dilakukan berdasarkan *database* yang dibuat oleh penulis.
e. *Stopwords,* yaitu Pada proses *stowords removal* ini proses yang dikerjakan yaitu menghapus kata yang kurang relevan atau kata yang tidak memiliki arti yang begitu penting dan berkaitan yang ada pada kalimat di dokumen dengan mencocokan *list* kata stopword yang pada *database* yang akan diuji. *Database* yang digunakan baik untuk Bahasa Indonesia maupun maupun Bahasa Inggris diambil dari sebuah situs yg menyediakan berbagain jenis *list stopword* untuk berbagai bahasa.

Pada umumnya tahap *preprocessing* terdapat 5 langkah yaitu *Case Folding, Filtering, Tokenizing, Stemming,* dan *Stopwords.* Ada satu langkah yang tidak penulis gunakan dalam pembuatan aplikasi peringkasan dokumen esktraktif ini yaitu proses *stemming. Stemming* yaitu proses menghilangkan imbuhan dari sebuah kata dan mengubahnya menjadi kata dasar. Tujuan dari penghilangan proses *Stemming* ini yaitu karena penulis memiliki pendapat bahwa dalam sebuah peringkasan dokumen ekstraktif imbuhan yang terdapat pada sebuah kata memiliki arti yang berbeda. Adapun contoh kalimat yaitu : “Bapak **membuatkan** sebuah rumah untuk istrinya”, “Bapak **dibuatkan** secangkir kopi oleh ibu”, “Bapak selalu **berbuat** baik kepada tetangga”. Kata “buat” pada kalimat pertama dan kalimat ketiga memiliki arti bapak melakukan sebuah kegiatan tapi pada kalimat dua kata”buat” memiliki arti bapak mendapatkan sebuah perlakuan. Apabila proses *stemming* tetap dilakukan maka tidak akan didapatkan perbedaan antara satu kata dengan yang lainnya.

Pada tahap filtering penulis menambahkan karakter “ ’s ” yang akan digunakan untuk menghapus tanda baca bagi dokumen yang menggunakan bahasa Inggris. Adapun contoh kalimatnya yaitu : “that is President’s car” berubah menjadi “that is President car”.

2.5 Term Frequency – Inverse Document Frequency (TF-IDF)

Metode *Term Frequency-Inverse Document Frequency* (TF-IDF) adalah cara pemberian bobot hubungan suatu kata (term) terhadap dokumen. Untuk dokumen tunggal tiap kalimat dianggap sebagai dokumen. Metode ini menggabungkan dua konsep untuk perhitungan bobot, yaitu *Term Frequency* (TF) merupakan frekuensi kemunculan kata (t) pada kalimat (s). *Document frequency* (DF) adalah banyaknya kalimat dimana suatu kata (t) muncul. Frekuensi kemunculan kata di dalam dokumen yang diberikan menunjukkan seberapa penting kata itu di dalam dokumen tersebut. [5]

Frekuensi kalimat yang mengandung kata tersebut menunjukkan seberapa umum kata yang ada pada dokumen uji tersebut. Bobot kata semakin besar jika sering muncul dalam suatu kalimat dan semakin kecil jika muncul dalam banyak kalimat. Pada Metode ini pembobotan kata dalam sebuah dokumen dilakukan dengan mengalikan nilai TF dan IDF. Pembobotan diperoleh berdasarkan jumlah kemunculan term dalam kalimat (TF) dan jumlah kemunculan term pada seluruh kalimat dalam dokumen (IDF). Bobot suatu istilah semakin besar jika istilah tersebut sering muncul dalam suatu dokumen dan semakin kecil jika istilah tersebut muncul dalam banyak dokumen. Nilai IDF sebuah term dihitung menggunakan persamaan (1) di bawah ini :

$$IDF = log\left(\frac{N}{Df}\right) \qquad (1)$$

keterangan :

N = Jumlah keseluruhan kalimat pada dokumen

Df = Jumlah kalimat yang mengandung keyword t

Menghitung bobot (W) masing-masing kalimat dengan persamaan (2) di bawah ini:

$$W_{d.t} = TF_{d.t} * IDF_t \qquad (2)$$

dengan :

d = kalimat ke-d

t = kata (*term*) ke-t

TF = *Term Frequency*

W = bobot kalimat ke-d terhadap kata (*term*) ke-t

IDF= *Inverse Document Frequency*

Kemudian baru melakukan proses pengurutan (sorting) nilai kumulatif dari W untuk setiap kalimat. Tiga kalimat dengan nilai W terbesar dijadikan sebagai hasil dari ringkasan atau sebagai output dari peringkasan teks otomatis. [5]

2.7 Model Analisis Dan Perancangan

Model analisis dan perancangan yang digunakan untuk membangun aplikasi ini adalah sebagai berikut:

2.7.1 *Unified Modeling Language* (UML)

Unified Modeling Language (UML) merupakan bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak. UML tidak berdasarkan pada bahasa pemrograman tertentu. Standar spesifikasi UML dijadikan standar *defacto* oleh OMG (*Object Management Group*) pada tahun 1997. UML yang berorientasikan obyek mempunyai beberapa notasi standar. Spesifikasi ini menjadi populer dan standar karena sebelum adanya UML, telah ada berbagai macam spesifikasi yang berbeda. Hal ini menyulitkan komunikasi antar pengembang perangkat lunak. Untuk itu beberapa pengembang spesifikasi yang sangat berpengaruh berkumpul untuk membuat standar baru. UML dirintis oleh Grady Booch, James Rumbaugh pada tahun 1994 dan kemudian Ivar Jacobson[6].

Menurut perintisnya, UML di definisikan sebagai bahasa visual untuk menjelaskan, memberikan spesifikasi, merancang, membuat model, dan mendokumentasikan aspek-aspek dari sebuah sistem. Karena tergolong bahasa visual, UML lebih mendepankan penggunaan diagram untuk menggambarkan aspek dari sistem yang sedang dimodelkan. Memahami UML itu sebagai bahasa visual itu penting, karena penekanan tersebut membedakannya dengan bahasa pemrograman yang lebih dekat ke mesin. Bahasa visual lebih dekat ke mental model pikiran kita, sehingga pemodelan menggunakan bahasa visual bisa lebih mudah dan lebih cepat dipahami dibandingkan apabila dituliskan dalam sebuah bahasa pemrograman.

Seperti yang telah dipaparkan di atas, UML yang merupakan turunan dan beberapa metode mempunyai kumpulan diagram *grafts* sebagai kombinasi dari konsep pemodelan data (*Entity Relationship Diagram*), pemodelan bisnis (*work flow*), pemodelan obyek, dan pemodelan komponen. Diagram *grafts* tersebut merupakan tampiian dari beberapa level abstraksi yang dapat digunakan secara bersama oleh semua proses pada seluruh *lifecycle* pengembangan perangkat lunak serta pada implementasi kebeberapa teknologi yang berbeda. UML memiliki berbagai jenis diagram, diantarnya adalah:

2.7.2 *Use case Diagram*

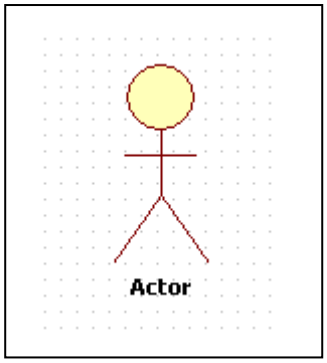
Use case diagram berisi mengenai interaksi antara sekelompok proses dengan sekelompok *actor*, menggambarkan fungsionalitas dari sebuah sistem yang dibangun dan bagaimana sistem berinteraksi dengan dunia luar[6]. *Use case diagram* dapat digunakan selama proses analisis untuk menangkap kebutuhan sistem dan untuk memahami bagaimana sistem seharusnya bekerja.

Use Case diagram terdiri dari beberapa elemen pemodelan utama, yaitu *Actor, Use Case, Association, Dependency,* dan *Generalization.*

a. *Actor*

Pada dasarnya *actor* bukanlah bagian dari *use case* diagram, namun untuk dapat terciptanya suatu *use case* diagram diperlukan beberapa *actor*. *Actor* tersebut mempresentasikan seseorang atau sesuatu (seperti perangkat, sistem lain) yang berinteraksi dengan sistem. *Actor* hanya berinteraksi dengan *use case*, tetapi

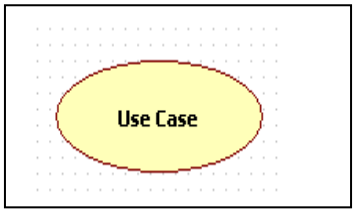
tidak memiliki kontrol atas *use case*. *Actor* digambarkan dengan stick man. Pada gambar 2.1 menunjukan Simbol *Actor* pada *Use Case* Diagram.



Gambar 2.1 Simbol *Actor* pada *Use Case* diagram

b. *Use Case*

Use case adalah gambaran fungsionalitas dari suatu sistem, sehingga customer atau pengguna sistem paham dan mengerti mengenai kegunaan sistem yang akan dibangun. Pada gambar 2.2 merupakan Simbol *Use Case* pada *Use Case Diagram.*



Gambar 2.2. Simbol *Use Case*

c. *Association*

Associaton menghubungkan *link* antar element, dan bukan menggambarkan aliran data / informasi pada sistem. *Association* digunakan untuk menggambarkan bagaimana *actor* terlibat dalam *use case.* Ada 4 jenis relasi yang bisa timbul pada *use case* diagram, yaitu *Association* antara actor dan *use case*, *Association* antara *use case*, Gener*alization/Inheritance antara use case, Generalization/Inheritance* antara *actors.*

d. *Dependency*

Dependency adalah sebuah *element* bergantung dalam beberapa cara ke *element* lainnya.

e. *Generalization*

Generalization disebut juga *inheritance* (pewarisan), sebuah elemen dapat merupakan spesialisasi dari elemen lainnya.

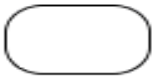
2.7.3 *Activity Diagram*

Activity diagram menggambarkan berbagai alir aktifitas dalam sistem yang sedang diarancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir[6]. *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 *behavior* internal sebuah sistem.

Terdapat berbagai simbol di dalam *Activity* Diagram, akan dijelaskan pada tabel 2.1

Tabel 2.1 Simbol *Activity* Diagram

Simbol	Nama	Keterangan
	*Actifity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain
	*Action*	*State* dari sistem yang mencerminkan eksekusi dari suatu aksi
	*Initial Node*	Bagaimana objek dibentuk atau diawali.
	*Actifity Final Node*	Bagaimana objek dibentuk dan dihancurkan
	*Fork Node*	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran

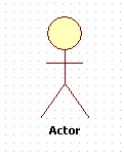
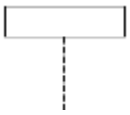
2.7.4 *Sequence Diagram*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang disusun dalam suatu urutan waktu[6].

Secara khusus, diagram ini berasosiasi dengan *use case*. *Sequence* diagram menggambarkan *behavior* internal sebuah sistem. Dan lebih menekankan pada penyampaian *message* dengan parameter waktu.

Pada Tabel 2.2 menjelaskan tentang simbol yang terdapat pada *Sequence* Diagram.

Tabel 2.2 Simbol *Sequence* Diagram

Simbol	Nama	Keterangan
	*Actor*	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi dan mendapat manfaat dari sistem dengan mengirimkan atau menerima pesan.
	*LifeLine*	Objek *entity*, antarmuka yang saling berinteraksi.
	*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi
	*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi

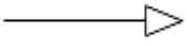
2.7.5 *Class Diagram*

Class Diagram adalah diagram yang menunjukan *class-class* yang ada dari sebuah sistem dan hubungannya secara logika dan menggambarkan struktur statis

dari sebuah sistem[6]. *Class* diagram digunakan untuk menampilkan beberapa kelas serta paket-paket yang ada dalam sistem/perangkat lunak yang sedang kita gunakan.

Class diagram memberi kita gambaran (diagram statis) tentang sistem/perangkat lunak dan relas-relasi yang ada didalamnya. Sebuah *class* digambarkan seperti sebuah bujur sangkar dengan tiga bagian ruangan yaitu bagian atas adalah bagian nama dari *class*, bagian tengah mendefinisikan *property* / *atribut class*, bagian akhir mendefinisikan *method-method* dari sebuah *class*. *Class* sebaiknya diberi nama menggunakan kata benda sesuai dengan domain / bagian / kelompoknya. Tabel 2.3 menjelaskan tentang *symbol relationship* pada *class* diagram:

Tabel 2.3 *Symbol Relationship Class* Diagram

Nama	Simbol	Keterangan
Asosiasi / *Association*		Merupakan sebuah *relationship* paling umum yang menghubungkan antara 2 *class*. Garis ini bisa melambangkan tipe-tipe *relationship* dan juga dapat menampilkan hukum-hukum multiplisitas pada sebuah *relationship*.(Contoh: *One-to-one*, *one-to-many*, *many-to-many*).
Asosiasi berarah / *Directed Association*		Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan multiplisitas
Generalisasi		Relasi antar kelas dengan makna generalisasi – spesialisasi (umum khusus)
Kebergantungan / *Dependency*		Umumnya penggunaan *dependency* digunakan untuk menunjukkan operasi pada suatu *class* yang menggunakan *class* yang

		lain.
Agregasi / *Agregation*		Relasi antar kelas dengan makna semua – bagian (*whole part*)

2.8 Teknik Evaluasi Sistem dan Pengujian

Adapun Teknik yang digunakan untuk mengevaluasi hasil suatu ringkasan teks merupakan topik yang cukup suilit, baik evaluasi terhadap ringkasan yang dihasilkan dari mesin peringkas otomatis ataupun ringkasan yang dihasilkan secara manual yang dibuat manusia, dikarenakan tidak terdapat definisi ringkasan ideal. Menurut Zaman B. dan E Winarko [12] metode untuk melakukan evaluasi terhadap hasil dari ringkasan secara umum dibagi 2, yaitu:

1. Ekstrinsik

Metode evaluasi ekstrinsik adalah menghitung efektivitas dan akseptabilitas dari hasil ringkasan untuk tugas-tugas tertentu, misalnya *assessment* terhadap hasil ringkasan.

2. *Intrinsik*

Metode evaluasi intrinsik adalah evaluasi yang dilakukan oleh sistem peringkas itu sendiri, misalnya menggunakan *F-Measures*. Evaluasi ini difokuskan pada tingkat koheren dan informatif dari hasil ringkasan.

Dalam penelitian ini, metode evaluasi yang digunakan adalah metode intrinsik penghitungan *F-Measure* berdasarkan perhitungan *Precision* dan *Recall* yang menurut Zaman B. dan E Winarko [12] merupakan standar evaluasi dalam penghitungan *information retrieval*. evaluasi perhitungan *information retrieval* dengan menggunakan *Precision* dan *Recall* juga dapat digunakan dalam evaluasi perhitungan peringkas teks otomatis.

$$Recall = \frac{\sum Kalimat\ Relevan\ yang\ dihasilkan\ Sistem}{\sum Kalimat\ ringkasan\ relevan} \quad (4)$$

$$Precision = \frac{\sum Kalimat\ Relevan\ yang\ dihasilkan\ Sistem}{\sum Kalimat\ yang\ dihasilkan\ sistem} \quad (5)$$

Kombinasi antara nilai *recall* dan *precision menghasilkan nilai* f-measure.

$$F - measure = \frac{2 * precision * recall}{recall + precision} \quad (6)$$

Dalam metode intrinsik, *precision* dan *recall* digunakan untuk mengukur kualitas ringkasan otomatis dengan cara membandingkan ringkasan otomatis dengan ringkasan manual (buatan manusia). Kemudian hasil akhir akan didapatkan dengan cara penggabungan nilai *recall* (4) dan *precision* (5) yang disebut dengan nilai *F-measures* (5). Masalah dalam metode ini adalah dalam menentukan kalimat relevan karena pasti terdapat perbedaan pendapat antar subyek pembuat ringkasan dalam memilih kalimat. Misal subyek A menganggap kalimat *x* sebagai kalimat utama tetapi bisa saja subyek B berpendapat sebaliknya bahwa kalimat *x* kurang begitu penting. Untuk mengatasi hal ini digunakan beberapa metode seperti suara terbanyak (*majority opinion*), gabungan (*union*), atau irisan (*intersection*)[12].

2.9 Perangkat Lunak Pendukung

Pada pembangunan aplikasi pencarian paragraf kata atau kalimat pada dokumen menggunakan *Term Frequency-Inverse Document Frequency* (TF-IDF) dibutuhkan perangkat lunak pendukung, diantaranya adalah:

2.9.1 Bahasa Program C#

Sebuah bahasa pemrograman yang berorientasi objek yang dikempbangkan oleh Microsoft sebagai bagian dari inisiatif kerangka .NET Framework. Bahasa pemrograman ini dibuat berbasiskan bahasa C++ yang telah dipengaruhi oleh aspek-aspek ataupun fitur bahasa yang terdapat pada bahasa-bahasa pemrograman lainnya seperti Java, Delphi, Visual Basic dll dengan beberapa penyederhanaan. Menurut standar ECMA-334 C# Language Specification, nama C# terdiri atas sebuah huruf latin C (U+0043) yang diikuti oleh tanda pagar yang menandakan angka #(U+0023). Tanda pagar yang

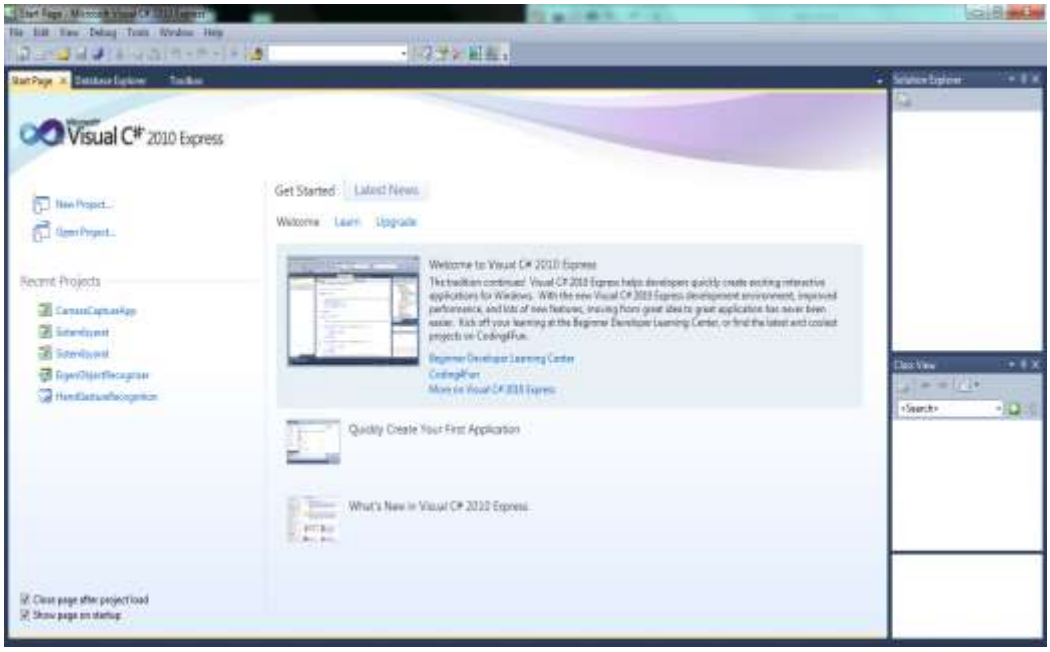
digunakan pada C# tidak sama dengan tanda kres yang ada dalam seni musik #(U+266F). Tanda pagar # (U+0023) tersebut digunakan karena karakter kres dalam seni musik tidak terdapat didalam keyboard standar.

C# kadang-kadang dapat disebutkan sebagai bahasa pemrograman yang paling mencerminkan dasar dari CLR dimana semua program-program .NET berjalan, dan bahasa ini sangat bergantung pada kerangka tersebut sebab ia secara spesifik di desain untuk mengambil manfaat dari fitur-fitur yang tersedia pada CLR. [7]

2.9.2 Microsoft Visual Studio

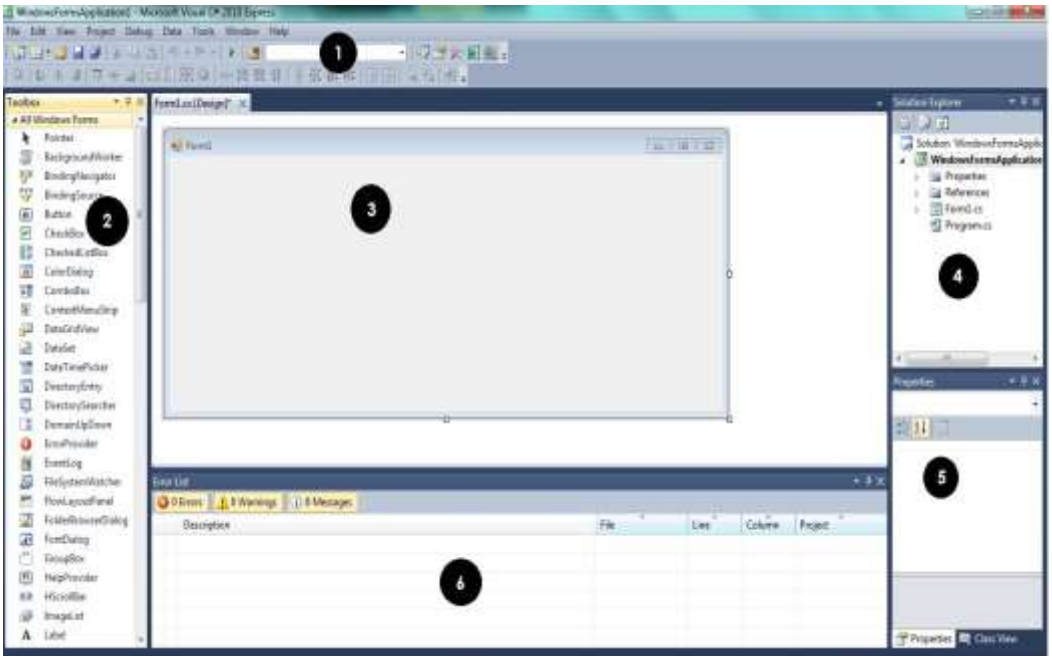
Microsoft visual stuido merupakan sebuah perangkat lunak lengkap (suite) yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal ataupun komponen aplikiasnya yang masuk ke dalam bentuk aplikasi console, Windows dan Web. Visual studio mencakup kompiler, SDK, Integrated Development Environment (IDE) dan dokumentsi (umumnya berupa MSDN Library). Kompiler yang dimasukan ke dalam paket visual studio antara lain Visual C++, Visual C#, Visual Basic, Visual Basic .NET, Visual InterDev, Visual J++, Visual J#, Visual FoxPro dan Visual SourceSafe.

Microsoft visual studio dapat digunakan untuk mengembangkan aplikasi dalam native code (dalam bentuk bahasa mesin yang berjalan diatas Windows) ataupun managed code (dalam bentuk Microsoft Intermediate Language diatas .NET Framework). Selain itu, visual studio juga dapat digunakan untuk mengembangkan aplikasi Silverlight, aplikasi Windows Mobile (yang berjalan diatas .NET Compact Framework). Berikut adalah tampilan halaman awal dari Microsoft Visual Studio yang dapat dilihat pada Gambar 2.3



Gambar 2.3 Halaman Utama Microsoft Visual Studio

Halaman awal tersebut merupakan tampilan utama untuk membuat suatu project baru, mmembuka project yang akan dilanjutkan pengerjaannya dan untuk membaca tutorial tentang bagaimana penggunaan aplikasi, cara untuk membuat aplikasi, produk terupdate yang dibuat oleh Microsoft Visual Studio dan pembelajaran coding.



Gambar 2.4 Tampilan Over View Microsoft Visual Studio

Pada Gambar 2.4 tampilan over view yang memperlihatkan tampilan antarmuka dari Microsof Visual Studio saat akan membuat seuah project, berikut adalah penjelasan sederhana tentang detail dari tampilan over view Microsoft Visual Studio :

1. *File Menu dan Toolbars Menu*

   Pada Gambar 2.4 nomor 1 dapat dilihat bagian ini terdiri dari fungsi-fungsi untuk melakukan aksi save, new project, add class, mendebug/execute program, search dan membuka project lain yang disimpan pada sistem directory.

2. *Toolbox Bar*

   Pada toolbox bar terdapat semua object yang dibutuhkan oleh pengguna yang tinggal dilakukan dengan proses drag and drop.

3. *Layout View*

   Layout view yaitu sebuah tempat atau wadah yang akan menampung object-object dari toolbox bar.

4. *Solutions Explorer Bar*

   Pada solution explorer bar terdapat program yang akan dibuat pengguna dan pada solution explorer juga bisa digunakan untuk menambahkan library-library.

5. *Properties Bar*

   Properties bar yaitu detail dari toolbox bar yang digunakan untuk mengedit object dari mulai penamaan object hingga warna object yang dipakai.

6. *Error List Bar*

   Pada error list bar terdapat pemberitahuan jika ada source code pengguna yang mengalami kesalahan.

2.9.3 *DevExpress*

DevExpress adalah *supplier* terkemuka kontrol ASP.Net yang *addins* digunakan dalam situs ASP.NET dan digunakan untuk menambah fitur, meringankan tugas dan menghemat waktu pemakai dalam membuat suatu program.

DevExpress menawarkan pula seperangkat komponen presentasi dan pelaporan serta menambahkan beberapa fungsi, karena itulah setiap situs *web* yang menggunakan ASP.Net banyak menggunakan aplikasi ini karena kemudahannya untuk memperindah tampilan tanpa memasukkan *coding* yang sangat banyak. [8]