BAB 2

TINJAUAN PUSTAKA

Pada bab ini dijelaskan beberapa konsep dan dasar teori yang berkaitan dengan permasalahan yang akan dibahas sebagai dasar pemahaman dalam mengimplementasikan konsep-konsep tersebut kedalam semua kegiatan pengembangan sistem yaitu profil sekolah, *learning management system*, algoritma koreksi *essay*, konsep perancangan sistem, dan pengembangan perangkat lunak.

2.1 Tinjauan Instansi

Pada tinjauan instansi ini akan dibahas mengenai sejarah, profil, visi dan misi SMK Negeri 1 Karawang.

2.1.1 Sejarah dan Profil SMK Negeri 1 Karawang

SMK Negeri 1 Karawang berdiri sejak 03 Agustus 1965 beralamat di JL. Pangkal Perjuangan (*By Pass*), Karawang berdasarkan SK. Pendirian No. 92/DIRPT/BI/65 yang ditandatangani oleh menteri pendidikan nasional. SMK Negeri 1 Karawang terdapat 9 jurusan keilmuan dan menjadi salah satu rintisan sekolah bertaraf internasional (RSBI) *Indonesia Vocational Education Strenghtening* (INVEST) dengan dasar dasar Surat Keputusan Direktur Jenderal Manajemen Pendidikan Dasar dan Menengah Departemen Pendidikan Nasional No : 10/C/KEP/MN/2009 tentang Penetapan 90 (Sembilan Puluh) SMK sebagai target dan sasaran pengembangan SMK-SBI melalui proyek *Indonesia Vocational Education Strenghtening* (INVEST).

Tabel 2.1 Profil SMK Negeri 1 Karawang

Nama Sekolah	SMK Negeri 1 Karawang
SKP	No.92/DIRPT/BI/65
Alamat Lengkap	Jl. Pangkal Perjuangan (*By Pass*) Karawang
Telp	0267401561
Tahun Berdiri	03 Agustus 1965
Terakreditasi	A

Peristiwa Penting:

1. Peresmian Gedung oleh Mendiknas pada tahun 1989.
2. Kerjasama dengan *Deutsche The Gesellschaft für Technische Zusammenarbeit* (GTZ) pada tahun 1996.
   a. Peningkatan kualitan layanan publik dalam bidang Pendidikan.
   b. Peningkatan *Link* and *match* Prakerin.
   c. Manajemen Bengkel.
3. Berstatus RSBI *Asian Development Bank* (ADB Invest) pada tahun 2009.
   a. Upaya meningkatkan mutu dan perluasan pendidikan di SMK, melalui Proyek Penguatan Pendidikan Menengah Kejuruan di Indonesia atau disebut *Indonesia Vocational Education Strengthening Project* (*INVEST*).
   b. Dana tersebut dimanfaatkan guna rehabilitasi atau perluasan bangunan, seperti ruang kelas, laboratorium, perpustakaan, dan kantor sekolah. Selain itu, dana proyek ini juga dimanfaatkan untuk membeli peralatan praktikum siswa, pelatihan guru, dan pengembangan kewirausahaan siswa.

2.1.2 Visi, Misi dan Tujuan Sekolah

1. Visi SMK Negeri 1 Karawang

Terwujudnya lembaga pendidikan dan pelatihan bertaraf internasional yang menghasilkan lulusan kreatif, inovatif dan profesional serta berdaya saing tinggi berlandaskan iman dan taqwa.

2. Misi SMK Negeri 1 Karawang

a. Menerapkan Sistem Manajemen Mutu ISO 9001:2008.
b. Melaksanakan proses pembelajaran berkarakter, berbasis teks, teknologi informasi dan komunikasi berwawasan lingkungan dan berkeunggulan lokal.
c. Menerapkan pendidikan dan pelatihan dengan sistem *Production Base Training* (PBT).
d. Melaksanakan pemelajaran bahasa asing yang berkualitas untuk menghadapi era globalisasi.
e. Menerapkan pendidikan dan pelatihan berbasis *Teaching Factory* bekerjasama dengan DU/DI.

3. Tujuan Strategis SMK Negeri 1 Karawang

SMK Negeri 1 Karawang Merupakan bagian dari pendidikan menengah dalam sistem Pendidikan Nasional, Bertujuan:

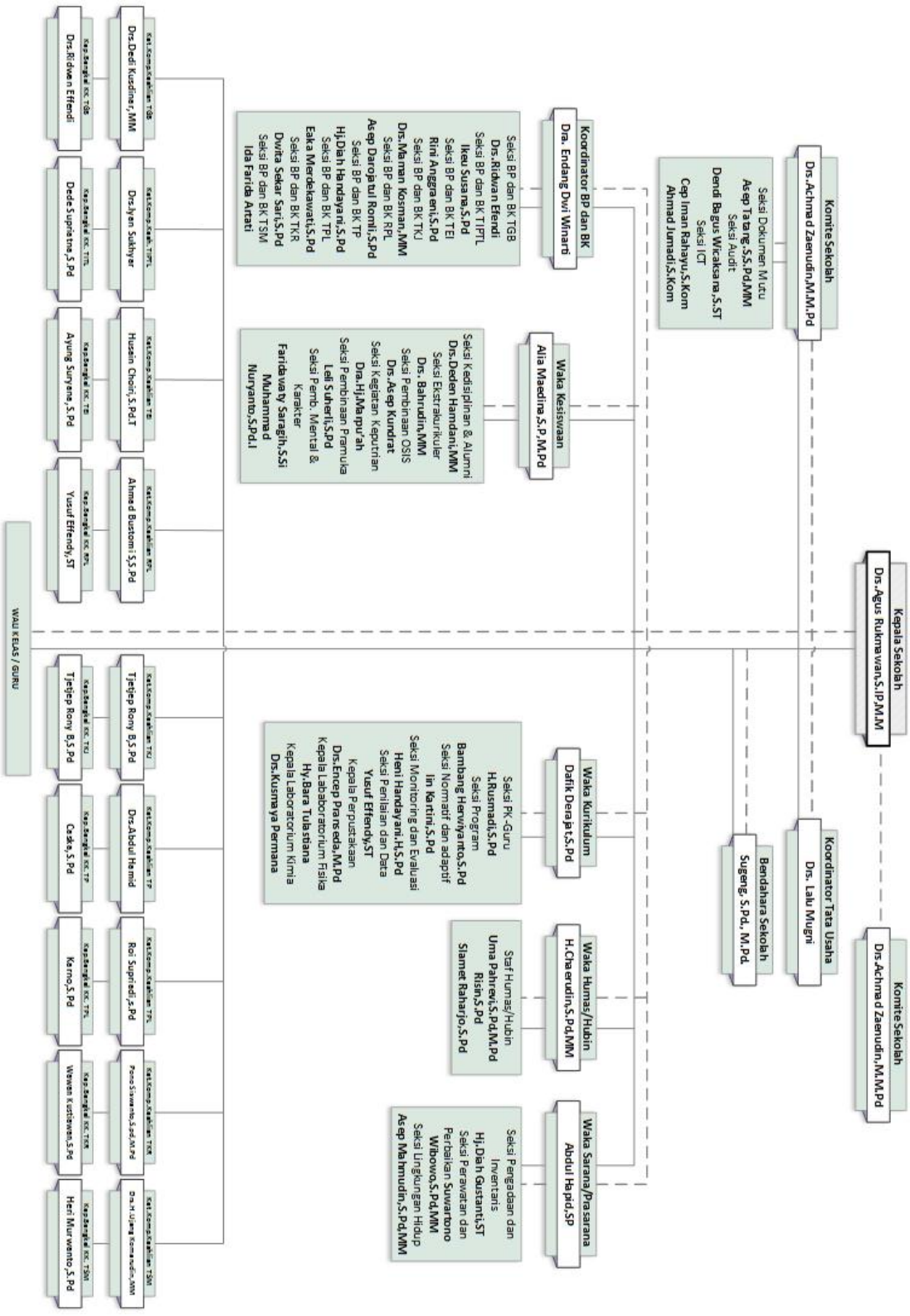
a. Mengutamakan penyiapan lulusan untuk memasuki dunia kerja serta mengembangkan sikap profesional.
b. Menyiapkan tenaga kerja tingkat menengah untuk mengisi kebutuhan dunia dunia usaha dan dunia industri.
c. Menyiapkan lulusan untuk membuka usaha mandiri atau berwirausaha.
d. Menyiapkan lulusan untuk mampu bersaing mengikuti Seleksi Masuk Perguruan Tinggi Negeri/Swasta bagi siswa yang ingin melanjutkan pendidikannya.

Semangat pengabdian SMK Negeri 1 Karawang kepada kehidupan yang berkualitas dinyatakan dalam moto REKAYASA “Religius, Berkarya, Berbudaya, dan Berwirausaha”.

2.1.3 Struktur Organisasi

Struktur organisasi menunjukan pola hubungan jalur komunikasi antara fungsi-fungsi, bagian-bagian atau orang-orang yang ada dalam suatu organisasi dapat diketahui bagaimana wewenang dan tanggung jawab mengalir diantara bagian tersebut.

Dibawah ini adalah bagan struktur organisasi staf pimpinan SMK Negeri 1 Karawang:



Gambar 2.1 Struktur Organisasi Staf Pimpinan SMK Negeri 1 Karawang

2.2 Landasan Teori

Dalam membangun *Learning Management System* untuk SMK Negeri 1 Karawang, terdapat landasan teori yang mendukung diantaranya:

2.2.1 *Learning Management System* (LMS)

Learning Management System (LMS) adalah system yang membantu administrasi dan berfungsi sebagai *platform e-learning content* [5]. Apabila memiliki banyak materi pelajaran *e-learning*, kita tidak hanya meletakkannya pada layar *desktop* komputer dalam bentuk *icon*. Bagaimana bila kita memiliki lebih dari 100 judul mata pelajaran *e-learning*? Layar komputer kita tentu akan penuh. Oleh karena itu, kita perlu memiliki LMS sebagai sistem yang mengatur *e-learning content* atau mata pelajaran *e-learning*. Berikut adalah fungsi LMS:

1. Katalog

LMS yang baik harus dapat menunjukkan materi pelajaran yang dimiliki. Materi-materi dapat berupa pelajaran *e-learning*, artikel, tesis, hasil diskusi dan lain sebagainya. Katalog yang baik pun harus dapat membedakan materi berdasarkan jenis materi, departemen yang memerlukan (misalnya, untuk *department marketing* dan penjualan) maupun kurikulum (misalnya, pelajaran anggaran untuk kurikulum keuangan, pelajaran *coaching* untuk kurikulum kepemimpinan). Katalog yang baik harus dapat menampilkan informasi tentang suatu pelajaran dengan lengkap, meliputi judul, tujuan, cakupan atau *outline*, durasi, target pelajar, tangggal tersedia, materi pendahuluan, tes yang harus diikuti dan lain sebagainya.

2. Registrasi dan persetujuan

Fungsi ini memungkinkan calon peserta pelatihan mendaftarkan diri secara *online*, baik untuk pelajaran *online* maupun di kelas. Apabila calon peserta pelatihan selesai mendaftar dan pelajaran membutuhkan persetujuan atasan atau orang lain, maka LMS harus menginformasikan kepada atasan (biasanya melalui *e-mail*) bahwa terdapat satu pendaftaran yang harus ditanggapi. Atasan dapat mengakses sistem dan melihat pendaftaran serta informasi tentang pelajar dan pelajaran. Kemudian, dia dapat mengambil tindakan, apakah disetujui atau

ditolak. LMS kembali menginformasikan hal tersebut kepada calon peserta pelatihan. LMS yang baik dapat pula menyimpan data pendaftaran dan persetujuan untuk membantu departemen pelatihan dalam memonitoring kegiatan *e-learning* di kemudian hari.

3. Menjalankan dan Memonitoring *E-learning*

LMS harus menjalankan materi pelajaran *e-learning* dengan baik. Apabila komputer pelajar tidak memiliki konfigurasi yang sesuai, maka LMS harus mengesannya dan memberikan peringatan kepada pelajar. Setelah materi pelajaran *e-learning* dijalankan, LMS harus mempunyai kemampuan merekam kegiatan agar dapat dibuat laporannya. LMS harus dapat merekam tentang berapa lama siswa mengakses materi pelajaran, berapa kali, tanggal dan jamnya dan lain sebagainya.

4. Evaluasi

LMS yang baik pun harus dapat melakukan bermacam evaluasi yang dapat mengukur keahlian peserta pelatihan sebelum dan sesudah mengikuti pelatihan. Evaluasi harus dapat mengukur seberapa jauh peserta pelatihan menyerap materi. Berdasarkan hasil evaluasi, LMS secara otomatis menyarankan untuk mengulang kembali pelajaran, membaca beberapa artikel tambahan, mengikuti pelajaran yang lain atau tindakan lainnya.

5. Komunikasi

LMS berguna pula sebagai sarana komunikasi bagi departemen pelatihan dan anggota organisasi. LMS dapat menyajikan atau memberikan pengumuman kepada pelajar tertentu. Pengumuman dapat dari pengajar atau administrator pelatihan. Komunikasi di sini dapat berarti pengajar memberikan materi bacaan tambahan kepada peserta pelatihan melalui sistem. Pengajar dapat pula memasukkan atau meng-*upload* sebuah artikel yang ditujukan pada beberapa pelajar tertentu dan LMS dapat menginformasikannya kepada mereka agar dapat mengakses dan men-*download* artikel melalui LMS.

6. Laporan

Melalui LMS, para administrator pelatihan dapat memperoleh laporan berisi data pelatihan. Atasan dan manajemen harus dapat mengakses system dan

mencetak laporan secara langsung, tanpa meminta bantuan administrator. Misalnya, seorang manajer ingin mengetahui berapa banyak anak buahnya yang telah memperoleh pelatihan MS *Powerpoint*. Ia mudah masuk atau *login* ke dalam LMS dan mencari data pelatihan anak buahnya.

7. Rencana Pembelajaran

Seorang manajer dapat membuat rencana pembelajaran untuk seberapa pelajar mengenai analisa kebutuhan belajar. Alangkah baiknya, bila rencana pembelajaran atau *training plan* dapat diikutsertakan dalam LMS. Jadi, berdasarkan rencana pembelajaran, LMS secara otomatis merekomendasikan program pembelajaran yang sesuai dan mengatur jadwalnya. Jadi, pelajar dapat melihat materi pembelajaran yang dia butuhkan melalui LMS, kapan dia bisa mengikuti dan harus menyelesaikan.

8. Integrasi

Dalam suatu organisasi, ada beberapa sistem komputer. LMS yang baik dapat berkomunikasi dan berintegrasi dengan sistem-sistem yang ada. Integrasi dengan sistem SDM adalah hal yang vital. Dengan integrasi yang baik, LMS akan mendapatkan daftar informasi karyawan terbaru dari sistem SDM yang sudah ada. Jadi, apabila pada sistem SDM terdapat perubahan data karyawan, penambahan karyawan baru atau karyawan yang keluar, maka informasi dapat digunakan untuk memperbaharui data pelatihan terkait.

Selain kedelapan fungsi dasar, kita dapat menambahkan fungsi-fungsi penunjang lain, misalnya forum diskusi atau *chatroom* agar pelajar dan pengajar dapat berdiskusi mengenai pembelajaran, dan lain-lain.

2.2.2 Metode dan Algoritma

2.2.2.1 *Text Preprocessing*

Text Preprocessing adalah langkah awal sebelum dilakukan sebuah pencocokan pada string. Tindakan yang dilakukan pada tahap ini adalah:

1. ***toLowerCase* / *case folding***

   yaitu mengubah semua karakter huruf menjadi huruf kecil.

2. ***Hapus delimiter (Stoplist)***

Menghilangkan delimiter-delimiter seperti tanda titik(.), koma(,), spasi dan karakter angka yang ada pada kata tersebut.

3. **Hapus kata umum (*stopword*).**

   Penghapusan kata umum seperti "adalah", "yaitu", dan lain-lain.

4. ***Tokenizing***

   pemotongan *string* dalam teks berdasarkan kata yang menyusunnya dan disimpan dalam sebuah variabel *array* [4].

2.2.2.2 *Latent Semantic Analysis* (LSA)

Latent Semantic Analysis (LSA) adalah suatu metode untuk mengekstrak sebuah tulisan dalam suatu dokumen dan kemudian mengaplikasikannya dalam perhitungan matematis. Penilaian dengan metode LSA lebih kepada kata-kata yang ada dalam tulisan tanpa memperhatikan urutan kata dan tata bahasa dalam tulisan tersebut, sehingga suatu kalimat yang dinilai adalah berdasarkan kata-kata kunci yang ada pada kalimat tersebut [2].

Untuk menghitung nilai jawaban *essay* menggunakan metode LSA, caranya adalah dengan cara membuat sebuah dokumen *essay* yang menjadi acuan (*query*) atau kunci jawaban dan kemudian *essay* yang akan dinilai dibandingkan dengan esai acuan jawaban, semakin banayak kemiripannya maka semakin besar nilai jawabannya.

Langkah-langkah pemeriksaan *essay* dengan metode LSA adalah pertama-tama tulisan direpresentasikan ke dalam matriks dimana baris matriks menunjukkan kata-kata kunci (*queries*) pada dokumen tersebut dan setiap kolom menunjukkan kolom dokumen. Setiap sel menunjukkan banyaknya kemunculan kata kunci yang berada pada matriks pada dokumen yang ada di kolom matriks. Kemudian isi dari sel tersebut terlebih dahulu ditransformasikan dimana setiap frekuensi kata dibobotkan dengan sebuah fungsi yang menunjukkan pentingnya sebuah kalimat dalam *paragraph* dan juga derajat yang menunjukkan seberapa pentingnya tipe kata didalam suatu kalimat.

Langkah selanjutnya adalah dilakukan dekomposisi *Singular Value Decomposition* (SVD) pada matriks kata-dokumen. Pada SVD matriks memuat frekuensi pemunculan kata kunci di dekomposisi menjadi tiga buah matriks yang jika tiga buah matriks tersebut dikalikan maka akan muncul kembali matriks asalnya. Matriks pertamanya mendeskripsikan entitas kolom sebagai nilai vektor orthogonal matriks. Matriks keduanya berupa matriks diagonal yang memuat nilai skalar matriks. Secara matematis, faktor yang paling baik adalah menggunakan dimensi terkecil dari matriks awalnya, sehingga rekonstruksi matriks terbaik dihasilkan pada saat nilai faktor lebih kecil dari jumlah faktor yang digunakan.

Dimensi dari matriks yang telah disederhanakan dengan menghapus koefisien pada matriks diagonal sebanyak yang diinginkan sampai tersisa koefisien sebanyak dimensi yang terpilih. Tujuan penyederhanaan ini adalah agar terbentuk matriks yang memuat nilai korelasi yang diinginkan ketika tiga buah matriks direkonstruksi. Kemudian penilaian akan dilakukan dengan membandingkan matriks korelasi dari *query* dengan matriks korelasi dari setiap dokumen dengan menggunakan perhitungan cosinus α.

2.2.2.3 Relevansi Dokumen Dan *Query*

Query dapat direpresentasikan sebagai vektor dalam ruang-*k*. vektor inilah yang kemudian dibandingkan dengan vektor-vektor dokumen untuk selanjutnya dinilai mana yang paling mendekati. Sebuah *query* seperti halnya dokumen, merupakan kumpulan dari kata-kata. *Query* pengguna dapat direpresentasikan sebagai:

$$\overline{d} = d^T U_k \sum_k^{-1} \quad \text{....(2-1)}$$

Mirip dengan vektor query, vektor dokumen direpresentasikan sebagai

$$\overline{d} = d^T U_k \sum_k^{-1} \quad \text{....(2-2)}$$

Matriks *q* adalah matriks satu kolom yang elemennya berisi nilai kehadiran kata kunci dalam query. Sementara matriks *d* adalah matriks satu kolom. Elemennya berisi nilai kehadiran kata kunci dalam dokumen. Matriks *d* sama dengan kolom matriks *A*.

$\bar{q}$ adalah vektor query dan $\bar{d}$ adalah vektor dokumen. Vektor query dapat dibandingkan dengan semua vektor dokumen yang ada. Teknik korelasi yang umum digunakan adalah dengan mencari nilai kosinus sudut yang dibentuk antara vektor query dan vektor dokumen. Korelasi kosinus antara vektor query dan vektor dokumen diberikan oleh persamaan

$$\cos \alpha = \frac{\bar{q}.\bar{d}}{\|\bar{q}\|\|\bar{d}\|} \quad ...(2\text{-}3)$$

α adalah sudut diantara kedua vektor tersebut. Jika $\bar{q}$ dan $\bar{d}$ dinormalisasi, maka magnitude dari vektor tersebut adalah 1 dan persamaan diatas dapat disederhanakan menjadi $\cos \alpha = \bar{q}.\bar{d}$. jadi, nilai korelasi adalah perhitungan sudut berdasarkan kosinus antara $\bar{q}$ dan $\bar{d}$. Jika dilakukan pengurutan dari dokumen yang paling dekat ke paling jauh relevansinya, maka dokumen yang paling dekat adalah dokumen yang memiliki sudut α dengan yang paling kecil.

2.2.3 Aplikasi Pendukung

Dalam perancangan sistem yang akan di bangun, diperlukan aplikasi pendukung sebagai sarana untuk mengimplementasikan sistem yang akan di bangun.

2.2.3.1 *Personal Homepage (PHP)*

PHP adalah kombinasi antara bahasa pemograman dan aplikasi *server* [11]. PHP merupakan *server-side* scripting yang digunakan untuk membuat dan menjalankan aplikasi web *server* yang dinamis, interaktif dan mempunyai performasi tinggi. Aplikasi *server* adalah program yang terdiri atas teknik-teknik dalam satu paket yang meliputi :

Ketangguhan bahasa pemograman

1. Pengaksesan basis data ke media penyimpanan yang permanen.
2. Mendukung *internet protocol*, khususnya HTTP dan *e-mail*.

PHP mampu berhubungan dengan basis data dan dapat diintegrasikan dengan HTML. PHP dapat berjalan dalam web server yang berbeda dalam sistem operasi yang berbeda pula. PHP ditulis dalam bahasa C, sehingga sebagian besar

sintak PHP mirip dengan C dan *perl*. Beberapa keuntungan menggunakan PHP adalah sebagai berikut :

1. *Open source*, semua *source* PHP tersedia.
2. PHP diterbitkan secara gratis, tidak ada biaya dalam mengimplementasikan.
3. *Cross-platform*, PHP dapat berjalan dalam sistem operasi *Windows* 98, *Windows* NT dan *Macintosh*.
4. PHP juga dapat berjalan pada server *Microsoft Personal Web Server*, IIS dan *Apache*.
5. *Embedded-html*, karena itu PHP mudah dipelajari.

2.2.3.2 *Database Management System* (DBMS)

DBMS (*Database Management System*) adalah suatu sistem aplikasi yang digunakan untuk menyimpan, mengelola dan menampilkan data. suatu sistem aplikasi disebut DBMS jika memenuhi persyaratan minimal sebagai berikut:

1. Menyediakan fasilitas untuk mengelola akses data.
2. Mampu menangani integritas data.
3. Mampu menangani akses data secara bersamaan.
4. Mampu menangani backup data.

DBMS sudah mulai berkembang sejak tahun 1960-an. Kemudian sekitar tahun 1970-an mulai berkembang teknologi *Relational* DBMS yaitu DBMS berbasis relasional model. Relasional model pertama kali dikembangkan oleh Edgar J. Codd pada tahun 1970. Secara sederhana relasional dapat dipahami sebagai suatu model yang memandang data sebagai sekumpulan tabel yang saling terkait. Hampir semua DBMS komersial dan *open source* saat ini berbasis Relasional DBMS.

Pada tahun 1980-an mulai berkembang *Object Oriented* DBMS (OODBMS). OODBMS berkembang seiring dengan perkembangan teknologi pemrograman berorientasi objek. Saat ini OODBMS juga cukup berkembang namun belum menggeser kepopuleran RDBMS.

MySQL adalah perangkat lunak DBMS atau server basis data yang dapat melayani banyak pengguna, dan dapat melakukan proses dalam waktu yang bersamaan (*multi threaded*). MySQL juga sangat cepat dan fleksibel dan dapat menyimpan data yang berkapasitas besar. Dalam perancangannya, MySQL mengutamakan kecepatan proses, ketangguhan dan kemudahan bagi penggunanya.

Pada saat ini MySQL adalah *server* basis data yang paling banyak diminati oleh banyak perusahaan di dunia. MySQL menyediakan lingkungan yang mudah dan efisien untuk menyimpan data dan mengakses basis data berkapasitas besar lebih cepat dari pada server basis data lainnya. Keuntungan lainnya adalah bahwa MySQL server tidak membutuhkan spesifikasi *hardware* yang mahal [10].

MySQL server memungkinkan membuat aplikasi *client* untuk mengakses dan memanipulasi data di *server* basis data melalui basis data *provider*. Keuntungan utama menggunakan MySQL *server* adalah mudah digunakan, berkecepatan tinggi dan membutuhkan sedikit memori serta tidak membutuhkan spesifikasi *hardware* yang mahal.

2.2.3.3 *Web Browser*

Browser web adalah *software* yang digunakan untuk menampilkan informasi dari *server web*. *Software* ini kini telah dikembangkan dengan menggunakan *user interface* grafis, sehingga pemakai dapat melakukan *point* dan *click* untuk pindah dokumen. Dapat dikatakan saat ini hanya ada 2 *browser* GUI yang popular, yaitu *Intenet Explorer* dan Netscape Navigator [9].

2.2.3.4 Pemodelan Analisis UML (*Unified Modelling Language*)

UML atau *Unified Modeling Language*, merupakan pemodelan berorientasi objek yang dikembangkan oleh Grady Bosch dan James Rumbaugh dari *Rational Software Corporation* pada akhir tahun 1994 dengan menggabungkan metode Booch dan OMT (*Object Modeling Technique*). Lalu pada tahun 1995, Ivar Jacobson dan perusahaan *Objectory* bergabung dengan Rational lalu menggabungkan metode OOSE (*Object Oriented Software*

Engineering) dan dari ketiga penggabungan tersebut lahirlah UML sebagai metode baru.

UML adalah salah satu *tool* / *model* untuk merancang pengembangan *software* yang berbasis *object oriented*. UML sendiri juga memberikan standar penulisan sebuah sistem *blue print,* yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema *database*, dan komponen komponen yang diperlukan dalam sistem *software*.

2.2.3.4.1 *Use Case Diagram*

Use case adalah rangkaian atau/uraian sekelompok yang saling terkait dan membentuk sistem secara teratur yang dilakukan atau diawasi oleh sebuah aktor.

Use case digunakan untuk membentuk tingkah laku benda dalam sebuah model dan direalisasikan oleh sebuah *colaboration*. Umunnya *use case* digunakan/ digambarkan dengan sebuah elips dengan garis yang solid, biasanya mengandung nama. *Use case* menggambarkan proses sistem (kebutuhan sistem dari sudut pandang *user*). *Use case* diagram terdiri dari :

1. *Actor*

Pada dasarnya aktor 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. Sebuah *actor* mungkin hanya memberikan informasi inputan pada sistem, hanya menerima informasi dari sistem atau keduanya menerima, dan memberi informasi pada sistem. *Actor* hanya berinteraksi dengan *use case,* tetapi tidak memiliki kontrol atas *use case*. Actor digambarkan dengan *stickman*. *Actor* dapat digambarkan secara secara umum atau spesifik, dimana untuk membedakannya kita dapat menggunakan *relationship*.

2. *Use Case*

Use case diagram adalah penggambaran sistem dari sudut pandang pengguna sistem tersebut (*user*), sehingga pembuatan *use case* lebih dititik-

beratkan pada fungsionalitas yang ada pada sistem, bukan berdasarkan alur atau urutan kejadian.

Tipe relasi/ *stereotype* yang mungkin terjadi pada *use case* diagram:

1. **<<i*nclude*>>**

   Yaitu kelakuan yang harus terpenuhi agar sebuah *event* dapat terjadi, dimana pada kondisi ini sebuah *use case* adalah bagian dari *use case* lainnya.

2. **<<e*xtends*>>**

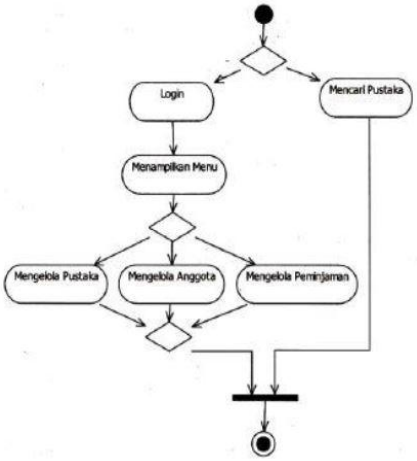
   Kelakuan yang hanya berjalan di bawah kondisi tertentu seperti menggerakkan alarm.

3. **<<*communicates*>>**

   Mungkin ditambahkan untuk asosiasi yang menunjukkan asosiasinya adalah *communicates association*. Ini merupakan pilihan selama asosiasi hanya tipe *relationship* yang dibolehkan antara *actor* dan *use case.*

2.2.3.4.2 *Activity Diagram*

Activity diagram memiliki pengertian yaitu lebih fokus kepada menggambarkan proses bisnis dan urutan aktivitas dalam sebuah proses. Dipakai pada *bussiness modeling* untuk memperlihatkan urutan aktifitas proses bisnis. Memiliki struktur diagram yang mirip *flowchart* atau data *flow diagram* pada perancangan terstruktur. Memiliki pula manfaat yaitu apabila kita membuat diagram ini terlebih dahulu dalam memodelkan sebuah proses untuk membantu memahami proses secara keseluruhan. Dan *activity* dibuat berdasarkan sebuah atau beberapa *use case* pada *use case diagram.*



Gambar 2.2 Contoh *Activity Diagram*

2.2.3.4.3 *Sequence Diagram*

Sequence diagram adalah suatu diagram yang memperlihatkan atau menampilkan interaksi-interaksi antar objek didalam sistem yang disusun pada sebuah urutan atau rangkaian waktu. Interaksi antar objek tersebut termasuk pengguna, *display*, dan sebagainya berupa pesan/ *message.*

Sequnce diagram digunakan untuk menggambarkan *scenario* atau rangkaian langkah-langkah yang dilakukan sebagai sebuah respon dari suatu kejadian/*event* untuk menghasilkan *output* tertentu. *Sequence diagram* diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

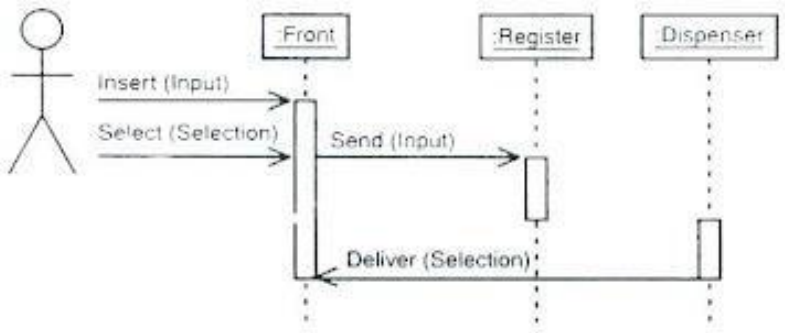
Sequence diagram biasanya digunakan untuk tujuan analisis dan desain, memfokuskan pada identifikasi *method* di dalam sebuah sistem.

Sequence diagram biasanya dipakai untuk memodelkan:

1. Deskripsi tentang sistem yang ada pada sebuah/ beberapa *use case* pada *use case diagram*, yang menggambarkan hubungan antara *actor* dan *use case diagram.*
2. Logika dari *method (operation, function, atau procedure).*
3. Logika dari *server (high level method).*

Tujuan penggunaan *sequence diagram*:

1. Mengkomunikasikan *requirement* kepada tim teknis karena diagram ini dapat lebih mudah untuk dielaborasi menjadi model *design*.
2. Merupakan diagram yang paling cocok untuk mengembangkan model deskripsi *use case* menjadi spesifikasi *design*.
3. Analisa dan desain, memfokuskan pada identifikasi *method* didalam sebuah sistem.



Gambar 2.3 Contoh *Sequence Diagram*

2.2.3.4.4 *Class Diagram*

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Class menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment* , pewarisan, asosiasi, dan lain-lain.

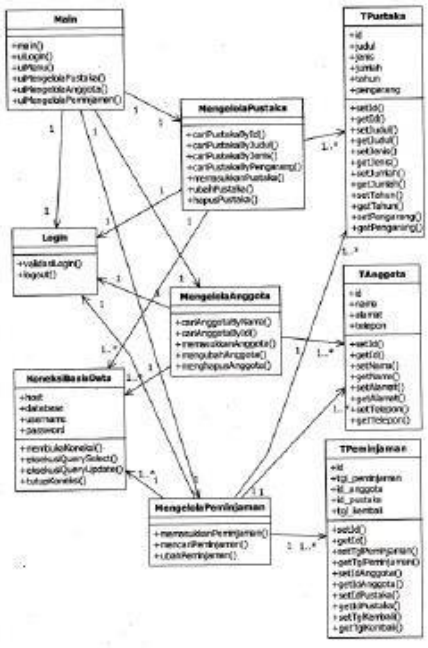
Class memiliki tiga area pokok :

1. *Boundary Class* adalah kelas yang memodelkan interaksi antara satu atau lebih *actor* dengan sistem. *Boundary* memodelkan bagian dari sistem yang bergantung pada pihak lain disekitarnya dan merupakan pembatas sistem dengan dunia luar.

2. *Control Class* digunakan untuk memodelkan "perilaku mengatur", khusus untuk satu atau beberapa *use-case* saja.
3. *Entity Class* memodelkan informasi yang harus disimpan oleh sistem. *Entity Class* memperlihatkan struktur data dari suatu sistem.

Hubungan Antar *Class*

1. Asosiasi, yaitu hubungan statis antar *class.* Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class.*
2. Agregasi, yaitu hubungan yang menyatakan bagian ("terdiri atas.").
3. Pewarisan, yaitu hubungan hirarkis antar class. Class dapat diturunkan dari *class* lain dan mewarisi semua atribut dan *metoda class* asalnya dan menambahkan fungsionalitas baru, sehingga disebut anak dari kelas yang diwarisinya. Kebalikan dari pewarisan adalah *generalisasi.*
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu kelas kepada kelas lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram.*



Gambar 2.4 Contoh *Class Diagram*

2.2.3.5 Pengujian Perangkat Lunak

Sebuah perangkat lunak perlu dijaga kualitasnya bahwa kualitas bergantung kepada kepuasan pelanggan (*customer*). Kualitas perangkat lunak perlu dijaga untuk keperluan berikut :

1. Agar dapat bertahan hidup di dunia bisnis perangkat lunak.
2. Dapat bersaing dengan perangkat lunak yang lain.
3. Penting untukpemasaran global (*global marketing*).
4. Mengefektifkan biaya agar tidak banyak membuang perangkat lunak karena kegagalan pemasaran atau kegagalan produksi.
5. Mempertahankan pelanggan (*customer*) dan meningkatkan keuntungan.

Sering perangkat lunak mengandung kesalahan (*error*) pada proses-proses tertentu pada saat perangkat lunak sudah berada ditangan *user*. Kesalahan tersebut sering disebut *bug*. Untuk menghindari banyaknya *bug* maka diperlukan adanya pengujian perangkat lunak sebelum perangkat lunak diberikan kepada pelanggan atau selama perangkat lunak masih dikembangkan[12].

Adanya *bug* adalah suatu yang biasa, bahkan di sebuah perangkat lunak yang terkenal dan besar pun biasanya masih ada *bug*, sehingga tidak perlu merasa tersinggung atau bersedih jika masih ditemukan *bug* pada perangkat lunak yang dikembangkan. Yang bisa dilakukan oleh pengembang perangkat lunak adalah meminimalisir *bug* dengan melakukan pengujian. Kelakuan perangkat lunak yang tidak sesuai dengan spesifikasi yang dibutuhkan bisa dianggap sebagai *bug*.

Pengujian dilakukan tidak hanya untuk meminimalisasi kesalahan secara teknis tapi juga kesalahan non teknis (misalnya pengujian pesan kesalahan sehingga *user* tidak bingung dengan pesan kesalahan yang muncul, dan juga jika masukan dan keluaran yang diperlukan berkapasitas sangat besar).

Pengujian perangkat lunak adalah elemen sebuah topik yang memiliki cakupan luas dan sering dikaitkan dengan verifikasi (*verification*) dan validasi (*validation*). Verifikasi mengacu pada sekumpulan aktifitas yang menjamin

bahwa perangkat lunak mengimplementasikan dengan benar sebuah fungsi yang spesifik. Validasi mengacu pada sekumpulan aktifitas yang berbeda yang menjamin bahwa perangkat lunak yang dibangun dapat ditelusuri sesuai dengan kebutuhan pelanggan (customer). Dapat dikatakan sebagai berikut :

1. Verifikasi : apakah produk yang dibangun dengan benar ? (lebih kearah apakah proses pengembangan produk sudah benar dan telah berhasil mengimplementasikan fungsi yang benar).
2. Validasi : apakah sudah membangun produk yang benar ? (lebih kearah hasil produk apakah sudah sesuai dengan yang diinginkan).

Verifikasi dan validasi meliputi beberapa aktifitas yang disebut sebagai jaminan kualitas perangkat lunak (*Software Quality Assurance* (SQA)). Verifikasi dan validasi mencakup lingkup yang luas dari aktifitas SQA yang didalamnya termasuk peninjauan teknis formal, audit kualitas dan konfigurasi, performansi, pengawasan, simulasi, studi kemungkinan, peninjauan dokumentasi, peninjauan basis data, analisis algoritma, pengujian proses pengembangan perangkat lunak, pengujian kualifikasi, dan pengujian instalasi. Untuk melakukan pengujian diperlukan pendefinisian kasus uji (*test case*) yang jelas.

Pengujian diawali dari pengujian unit. Unit disini bisa berupa kumpulan fungsi atau prosedur yang memiliki keterkaitan pada pemrograman terstruktur (misalkan unit untuk menuliskan atau membaca data pada basis data) atau kelas pada pemrograman berorientasi objek. Unit juga dapat berupa modul atau dikenal sebagai *package*. Setelah unit selesai diuji maka dilakukan pengujian integrasi.

Pengujian integrasi sebagiknya dilakukan secara bertahap, tidak dilakukan secara satu tahap langsung di akhir untuk menghindari kesulitan penelusuran jika terjadi kesalahan (*error*). Pengujian integrasi lebih kepada pengujian penggabungan dari dua atau lebih unit pada perangkat lunak. Setelah pengujian integrasi maka dilakukan pengujian sistem dimana unit-unit proses yang sudah diintegrasi diuji dengan antarmuka yang sudah dibuat sehingga pengujian ini dimaksudkan untuk menguji sistem perangkat lunak secara keseluruhan dan diuji secara satu sistem.

Setelah pengujian sistem selesai dilakukan maka dapat dilakukan pengujian penerimaan perangkat lunak oleh pelanggan (*customer*) atau *user*. Pengujian penerimaan digunakan untuk mengetahui kepuasan pelanggan atau *user* terhadap perangkat lunak yang sudah dibuat. Jika pelanggan sudah puas dengan perangkat lunak, maka perangkat lunak dapat diserahkan kepada pelanggan (*customer*).

Pengujian untuk validasi memiliki beberapa pendekatan sebagai berikut:

1. ***Black-Box testing*** **(Pengujian Kotak Hitam)**

Menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dank ode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan.

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

a. Jika *user* memasukan nama pemakai (*username*) dan kata sandi (*password*) yang benar.
b. Jika *user* memasukan nama pemakai dan kata sandi yang salah, misalnya nama pemakai benar tapi kata sandi salah, atau sebaliknya, atau keduanya salah.

2. ***White-Box Testing*** **(Pengujian Kotak Putih)**

Menguji perangkat lunak dari segi desain dan kode program apakah mampu menghasilkan fungsi-fungsi, masukan dan keluaran yang sesuai dengan spesifikasi kebutuhan. Pengujian kotak putih dilakukan dengan memeriksa lojik dari kode program. Pembuatan kasus uji bisa mengikuti standar pengujian dari standar pemrograman yang seharusnya. Contoh dari pengujian kotak putih misalkan menguji alur (dengan menelusuri) pengulangan (*looping*) pada logika

pemrograman. Pengujian terhadap dokumentasi yang dibuat juga harus dilakukan agar dokumentasi yang dibuat tetap konsisten dengan perangkat lunak yang dibuat.

2.2.3.6 Teknik Pengumpulan Data

Hal pertama yang dilakukan dalam analisis sistem adalah melakukan pengumpulan data. Ada beberapa teknik pengumpulan data yang sering dilakukan yaitu sebagai berikut :

1. Teknik Wawancara

Pengumpulan data dengan menggunakan wawancara mempunyai beberapa keuntungan sebagai berikut:

a. Lebih mudah dalam menggali bagian sistem mana yang dianggap baik dan bagian mana yang dianggap kurang baik.
b. Jika ada bagian tertentu yang menurut anda perlu untuk digali lebih dalam, anda dapat langsung menanyakan kepada narasumber.
c. Dapat menggali kebutuhan *user* secara lebih bebas.
d. *User* dapat mengungkapkan kebutuhannya secara lebih bebas.

Selain mempunyai beberapa kelebihan tersebut, teknik wawancara juga mempunyai beberapa kelemahan. Berikut ini adalah beberapa kelemahan dari teknik wawancara.

a. Wawancara akan sulit dilakukan jika narasumber kurang dapat mengungkapkan kebutuhannya.
b. Pertanyaan dapat menjadi tidak terarah, terlalu fokus pada hal tertentu dan mengabaikan bagian lainnya.

2. Teknik Observasi

Pengumpulan data dengan menggunakan observasi mempunyai keuntungan yaitu :

a. Analis dapat melihat langsung bagaimana sistem lama berjalan

b. Mampu menghasilkan gambaran lebih baik jika dibanding dengan teknik lainnya.

Sedangkan kelemahan dengan menggunakan teknik observasi adalah :

a. Membutuhkan waktu cukup lama karena jika observasi waktu sangat terbatas maka gambaran sistem secara keseluruhan akan sulit diperoleh.
b. Orang yang sedang diamati biasanya berperilaku berbeda dengan sehari-hari (cenderung berusaha terlihat baik). Hal ini menyebabkan gambaran yang diperoleh selama observasi akan berbeda dengan perilaku sehari-hari.
c. Dapat mengganggu pekerjaan orang-orang pada bagian yang sedang diamati

Berikut ini adalah beberapa petunjuk untuk melakukan observasi

a. Tentukan hal-hal apa saja yang akan diobservasi agar kegiatan observasi menghasilkan sesuai dengan yang diharapkan.
b. Mintalah izin kepada orang yang berwenang pada bagian yang akan diobservasi.
c. Berusaha sesedikit mungkin untuk tidak mengganggu pekerjaan orang lain.
d. Jika ada yang tidak dimengerti, cobalah bertanya. Jangan membuat asumsi sendiri.

3. Teknik Kuisioner

Pengumpulan data dengan menggunakan kuisioner mempunyai keuntungan yaitu :

a. Hasil lebih objektif, karena kuisioner dapat dilakukan kepada banyak orang sekaligus
b. Waktu lebih singkat.

Sedangkan kelemahan pengumpulan data dengan menggunakan kuisioner adalah sebagai berikut.

a. Responden cenderung malas untuk mengisi kuisioner.

b. Sulit untuk membuat pertanyaan yang singkat, jelas, dan mudah dipahami.

Berikut adalah beberapa cara yang dapat dilakukan untuk membuat teknik kuisioner menghasilkan data yang baik:

a. Hindari pertanyaan isian, lebih baik pilihan ganda karena responden biasanya malas untuk menulis banyak dan jika responden menuliskan sesuatu sering kali susah untuk dipahami, dan juga dengan pertnayaan pilihan ganda, akan memudahkan anda untuk melakukan rekapitulasi data hasil kuisioner.
b. Buatlah pertanyaan yang tidak terlalu banyak
c. Buatlah pertanyaan yang singkat, padat dan jelas.

2.2.3.7 Teknik Pengujian Akurasi Sistem

Skenario pengujian ini dimaksudkan untuk melihat seberapa besar galat yang ada pada sistem saat aplikasi dijalankan. Untuk mengetahui rentang galat antara penilaian sistem dan penilaian secara manual, maka digunakan rumus :

Jika nilai guru lebih besar dari nilai sistem, maka rumusnya adalah :

Persentase galat (%) = 100% - (Nilai Sistem / Penilaian Manual) x 100% **[15]**

Jika nilai guru lebih kecil dari nilai sistem, maka rumusnya adalah :

Persentase galat (%) = 100% - (Penilaian Manual / Nilai Sistem) x 100% **[15]**

Hal yang menjadi penyebab perbedaan standar penilaian yang dilakukan antara sistem dan penilaian secara manual adalah :

a. Pada penilaian manual, jika sudah ada kalimat jawaban yang benar pada jawaban, maka jawaban akan mendapat nilai tertinggi. Sedangkan pada penilaian sistem, panjang pendek kalimat jawaban dan kata yang pada kalimat jawaban akan sangat berpengaruh pada nilai yang diberikan.
b. Penyebab lain adalah karena penilaian sistem sangat berpengaruh pada kelengkapan jawaban secara keseluruhan.

c. Penyebab lain yang terjadi adalah apabila kalimat jawaban memiliki susunan kalimat yang berbeda dengan kunci jawaban maka akan tetap dinilai sama.