BAB II

LANDASAN TEORI

2.1 Pengertian Aplikasi

Aplikasi menurut Jogiyanto (1999:12), adalah penggunaan dalam suatu komputer, instruksi (*instruction*), atau pernyataan (*statement*) yang disusun sedemikian rupa sehingga komputer dapat memproses input menjadi output. Menurut *Kamus Besar Bahasa Indonesia* (1998:52), aplikasi adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu.

Aplikasi adalah suatu program komputer yang dibuat untuk mengerjakan dan melaksanakan tugas khusus dari pengguna. Aplikasi merupakan rangkaian kegiatan atau perintah untuk dieksekusi oleh komputer.

Program merupakan kumpulan *instruction set* yang akan dijalankan oleh pemroses, yaitu berupa *software*. Bagaimana sebuah sistem komputer berpikir diatur oleh program ini. Program inilah yang mengendalikan semua aktivitas yang ada pada pemroses. Program berisi konstruksi logika yang dibuat oleh manusia dan sudah diterjemahkan ke dalam bahasa mesin sesuai dengan format yang ada pada *instruction set*.

Program aplikasi merupakan program siap pakai. Program yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain. Contoh dari aplikasi adalah program pemroses kata dan Web Browser. Aplikasi akan menggunakan sistem operasi (OS) komputer dan aplikasi lain yang mendukung. Istilah ini mulai perlahan masuk ke dalam istilah Teknologi Informasi semenjak

tahun 1993, yang biasanya disingkat dengan istilah "*app*". Secara historis, aplikasi adalah *software* yang dikembangkan oleh sebuah perusahaan. *App* adalah *software* yang dibeli perusahaan dari tempat pembuatnya. Industri PC tampaknya menciptakan istilah ini untuk merefleksikan medan pertempuran persaingan yang baru, yang paralel dengan yang terjadi antar sistem operasi yang dimunculkan.

2.2 Pengertian Visualisasi

Visualisasi adalah rekayasa dalam pembuatan gambar, diagram, atau animasi untuk penampilan suatu informasi. Saat ini visualisasi seringkali digunakan dalam keperluan ilmu pengetahuan, rekayasa desain produk, pendidikan, kedokteran, hingga kegiatan medis yang memerlukan visualisasi dalam penyampaiannya.

Aplikasi visualisasi dapat diartikan sebagai perangkat lunak yang dibuat untuk mempermudah penggunanya dalam mengolah informasi dalam bentuk gambar, diagram, maupun animasi.

2.3 Pengertian Fitness dan Physical Fitness

Fitness menurut *Phoenix Advance Dictionary* (2008:149) adalah kebugaran. Menurut Sharon Stoll, Ph.D (1986:1) secara istilah *fitness* merupakan aktivitas yang mampu membuat orang menjadi lebih bugar dengan menjalankan tiga komponen utamanya, yaitu olahraga, pemenuhan nutrisi dan isitirahat. Ruang lingkup fitness termasuk didalamnya latihan angkat beban (*weight training*) dan aerobik (*aerobics*) yang berguna untuk meningkatkan profil *hormone* dan *libido*, meningkatkan imunitas tubuh terhadap penyakit, mencegah pengapuran pada tulang dan banyak lagi keuntungan lain yang didapat dari olahraga *fitness*.

Secara harfiah *physical fitness* berarti kesesuaian fisik atau kecocokan jasmani. Hal yang harus sesuai dan dipadankan adalah tugas-tugasnya, yang dalam pelaksanaannya tergantung dari aspek-aspek jasmaniah dan rohaniah individu yang bersangkutan. Misalnya, untuk mencapai prestasi yang setinggi-tingginya diperlukan kesesuaian bentuk tubuh cabang olahraga yang bersangkutan, efektivitas organ-organnya untuk cabang olahraga tersebut, dan pandangan atau sikap individu untuk melaksanakan kegiatan tersebut sehingga muncul istilah *anatomical fitness*, *physiological fitness*, dan *psychological fitness*.

Seseorang dikatakan mempunyai *anatomical fitness* untuk melakukan suatu usaha/kegiatan apabila ia memenuhi persyaratan kelengkapan anggota-anggota tubuh yang diperlukan untuk melakukan kegiatan itu. Seseorang dikatakan mempunyai *physiological fitness* untuk melakukan suatu kegiatan apabila ia dapat melakukannya dengan tangkas dan dapat pulih (*recovery*) kembali dengan cepat dari keadaan yang timbul sebagai akibat kegiatan tersebut. Semua kegiatan memerlukan kekuatan otot, ketangkasan, dan daya tahan, walaupun tidak sama untuk bermacam-macam kegiatan. Secara singkat *physiological fitness* adalah kemampuan tubuh untuk berfungsi secara optimal. Seseorang dikatakan mempunyai *psychological fitness* untuk melakukan suatu kegiatan apabila ia mempunyai sifat-sifat mental yang diperlukan. Misalnya, kemauan yang besar yang memungkinkan untuk mengatasi atau tidak menghiraukan rasa yang tidak menyenangkan, rasa sakit, dan sebagainya sebagai akibat dari berlangsungnya kegiatan tersebut.

2.4 Pengertian Weight Training

Weight Training atau latihan beban adalah bagian dari olahraga *fitness*. Latihan ini dapat dilakukan menggunakan alat bantu seperti *dumbbell*, *barbell*, *bench press machine* ataupun menggunakan badan sendiri sebagai bebannya seperti latihan *sit-up*, *push-up*, *pull-up*. Menurut Harsono (1988:55) *weight training* adalah latihan-latihan yang sistematis dimana beban hanya dipakai sebagai alat untuk menambah tahanan terhadap kontraksi otot guna mencapai berbagai tujuan tertentu, seperti untuk meningkatkan dan menjaga kondisi fisik, kesehatan, kekuatan atau prestasi dalam suatu cabang olahraga tertentu. Beberapa syarat dan prinsip penting yang harus diperhatikan dalam melaksanakan latihan weight training antara lain:

1. *Weight training* harus didahului oleh gerakan pemanasan (*warming up*) secara menyeluruh.
2. Prinsip beban lebih (*overload*) yaitu kondisi diantara kemampuan untuk menggunakan beban yang maksimal dan kondisi ketidakmampuan mengangkat beban, harus diterapkan.
3. Membuat patokan atau kriteria dalam jumlah berat beban, pengulangan (*repetisi*), *set* yaitu jumlah periode dari satu gerakan, dan istirahat (*recovery*) untuk beberapa jenis latihan tertentu, seperti untuk latihan kekuatan (*strength*), daya tahan (*endurance*) dan *power*, patokan atau kriterianya berbeda.
4. Setiap mengangkat, mendorong atau menarik beban harus dilaksanakan dengan teknik atau cara yang benar dan sungguh-sungguh.

5. Repetisi sedikit dengan beban maksimum akan membentuk kekuatan (*strength*), sedangkan repetisi banyak (kira-kira 15–20 repetisi) dengan beban ringan atau sedang akan menghasilkan perkembangan daya tahan (*endurance*), dan repetisi sedang dengan beban sedang atau berat dalam jumlah yang sedang atau rendah diikuti dengan percepatan ketika melakukannya maka akan menghasilkan tenaga (*power*).
6. Setiap bentuk latihan harus dilakukan dalam ruang gerak seluas-luasnya.
7. Selama latihan atau mengangkat beban, pengaturan pernapasan harus diperhatikan. Dalam pengaturan pernapasan sebaiknya melakukan hal-hal berikut (1) pada waktu mengangkat beban atau bagian terberat dari mengangkat beban lakukan pengambilan napas (inhalasi), (2) pada waktu beban sudah mulai diturunkan atau bagian ringan dari angkat beban lakukan pengeluaran napas (exhalasi). Perlu diingat bahwa jangan menahan napas ketika mengangkat atau menurunkan beban.
8. Pada akhir melakukan suatu bentuk latihan, seseorang harus berada dalam keadaan yang lelah. *Weight training* sebaiknya dilakukan tiga kali dalam seminggu. Hal ini dimaksudkan untuk memberikan kesempatan kepada metabolisme otot untuk beristirahat diantara selingan hari dalam seminggu tersebut.

Metode dan sistem latihan *weight training* terdiri dari:

1. Set System

Pelaksanaannya yaitu melakukan beberapa repetisi dari suatu bentuk latihan, diiringi dengan istirahat sesaat, untuk kemudian mengulangi lagi repetisi seperti semula. Banyak para ahli menyatakan bahwa perkembangan kekuatan

otot akan lebih cepat apabila atlet berlatih sebanyak 3 set dengan 8-12 RM (Repetisi Maksimal), yaitu kondisi dimana seseorang sanggup mengangkat beban terberat selama satu kali dengan sempurna, untuk setiap bentuk latihan. Untuk daya tahan otot bisa dilakukan 20-25 RM, sedangkan untuk melatih *power* bisa dilakukan 12-15 RM. Dan sebaiknya dilakukan 3 kali dalam seminggu agar pada hari-hari tanpa latihan dapat dikondisikan untuk pemulihan otot dari kelelahan. Sebagai landasan tambahan, hasil penelitian dari Delorme dan Watkins (Bowers dan Fox, 1992) menjelaskan bahwa program latihan kekuatan otot (kontraksi isotonik) terdiri dari 1-3 *set* dengan menggunakan beban dan melakukan repetisi 2-10 RM. Dan apabila pelaksanaan 6 *set* dengan beban RM yang tinggi akan membutuhkan banyak waktu. Pada program latihan yang disusun oleh Delorme dan Watkins ini, frekuensi latihan 4 kali dalam seminggu merupakan batas maksimal yang dapat ditolelir. Selanjutnya para pelatih telah sepakat bahwa latihan 3 kali dalam seminggu akan meningkatkan kekuatan tanpa ada resiko yang kronis. Perlu ditekankan bahwa kelelahan yang kronis yang disebabkan kurangnya istirahat merupakan hal yang harus dihindari. Lebih jelas lagi, istirahat disini bukan hanya dibutuhkan per hari tapi juga antara *set* yang satu dengan *set* yang lainnya. Jika frekuensi latihan diperhatikan, maka pencapaian kekuatan yang signifikan dapat terjadi setelah 6 minggu berikutnya, atau lebih lama dari itu.

2. Super Set

   Pelaksanaannya yaitu setiap bentuk latihan diteruskan dengan bentuk latihan untuk otot-otot yang saling berlawanan. Program latihan ini sangat

melelahkan, karena disamping harus melakukan bentuk latihan otot-otot bagian depan (diagonis) juga harus melakukan bentuk latihan otot-otot bagian belakang tubuh (antagonis).

3. Split Routines

   Pelaksanaannya yaitu setiap bentuk latihan dibagi-bagi dalam setiap harinya. Misalnya hari ini melatih bagian atas tubuh, dan pada hari berikutnya melatih bagian-bagian tubuh bawah. Program ini dilakukan, apabila waktu untuk berlatih sangat terbatas.

4. Sistem Multi Poundage

   Pelaksanaannya yaitu setiap bentuk latihan dimulai dengan melakukan beberapa repetisi dengan beban yang berat. Kemudian, bila tampak tanda-tanda kelelahan mulai timbul dan hampir tidak dapat lagi mengangkat beban yang berat itu (misalnya pada repetisi ke-5 atau 6), segera salah seorang teman (*partner*) mengurangi berat beban tersebut dengan mencopot beberapa beban, sedangkan seseorang yang sedang melakukan masih terus mengangkat beban tanpa istirahat (sampai jumlah repetisi yang sesuai). Bila kemudian timbul tanda-tanda kelelahan lagi, temannya dapat mengurangi lagi beban yang sedang diangkat dan begitu seterusnya sampai 20 repetisi.

5. Sistem Burn-Out

   Sistem latihan ini sangat berat pelaksanaannya, karena disamping menekankan pada kekuatan, juga pada daya tahan otot, maka dari itu otot-otot harus bekerja sampai tenaganya habis (*burned-out*). Dan urutan bentuk latihannya pun bergantian, mulai anggota tubuh bagian atas sampai anggota tubuh bagian bawah. Pelaksanaannya adalah sebagai berikut (1) berat beban

hanya mampu diangkat satu kali atau 1 RM, (2) kemudian beban dikurangi sampai hanya bisa diangkat 2 RM, (3) selanjutnya beban dikurangi lagi sampai hanya mampu mengangkat beban 3 RM adalah angkatan yang maksimal, (4) demikian seterusnya sampai seseorang tidak mampu lagi mengangkat beban karena kehabisan tenaga, yang berarti tenaga mereka telah terbakar habis, (5) istirahat diantara *set* adalah 5 detik, waktu tersebut hanya cukup untuk mencopot besi dari tiangnya. Pada metode ini tidak membatasi jumlah repetisi angkatannya. Sehingga, sebaiknya menggunakan alat *weight training machine* yang bobotnya lebih mudah untuk dikurangi. Sistem ini pun metodenya hampir sama dengan Sistem *Multi Poundage*.

6. Sistem Piramid

   Pelaksanaan Sistem Piramid adalah sebaliknya dari Sistem *Burn Out*. Yaitu beban untuk set 1 ringan, kemudian pada set-set berikutnya semakin lama semakin berat. Dan biasanya jumlah *set* dalam Sistem Piramid dibatasi sampai 5 *set*. Istirahat antara *set* adalah 3-5 menit.

2.5 Pengertian Aerobics

Menurut Sharon Stoll, Ph.D (1986:2) *aeobics* (aerobik) mengacu pada setiap kegiatan yang menggunakan otot-otot besar untuk merangsang jantung dan paru-paru dalam suatu jangka waktu tertentu untuk merangsang pergerakan *cardiovascular* (otot jantung). Contoh kegiatan aerobik adalah berjalan, berlari, melompat tali, mendaki, bersepeda, dll. Namun, banyak kegiatan yang seharusnya termasuk dalam kegiatan aerobik yang tidak dapat dilakukan jika kegiatan tersebut tidak melibatkan intensitas tertentu. Dengan kata lain, jika detak jantung

anda turun di bawah tingkat *Heart Rate*/HR (detak jantung) minimum anda dalam beraktivitas, maka anda tidak dikategorikan sedang melakukan kegiatan aerobik.

Anda harus menjaga detak jantung anda pada efek bekerja dengan durasi minimum selama 20 sampai 30 menit anda untuk mendapatkan efek dari aerobik. Setiap kegiatan yang tidak meningkatkan detak jantung anda dalam efek beraktivitas dan tidak berada pada suatu durasi tertentu, disebut tidak aerobik. Bahkan aktivitas berjalan dapat dikategorikan bukan aerobik jika anda berjalan pada kecepatan yang tidak memacu detak jantung anda untuk efek bekerja dan anda terus-menerus berhenti atau beristirahat di bawah detak jantung minimum anda.

Berdasarkan sifat dan kelasnya, secara garis besar aerobik dibagi dalam tiga macam yaitu:

1. Low Impact

   Low Impact adalah gerakan aerobik yang dilakukan dengan intensitas rendah, antara lain dengan hentakan-hentakan ringan dalam posisi kaki tetap di lantai, misalnya pada saat *jogging* (lari-lari kecil) kaki tidak terangkat tinggi. Cara ini biasanya dilakukan oleh kalangan pemula, usia lanjut, dan orang yang mengalami obesitas (kelebihan berat badan).

2. High Impact

   High Impact adalah gerakan aerobik yang dilakukan intensitas tinggi, biasanya untuk memicu *cardiovascular* (otot jantung). Cara ini biasanya dilakukan oleh mereka yang sudah terbiasa dengan latihan aerobik dan *cardio training*.

3. Mix Impact

Mix Impact adalah gerakan aerobik yang mengkombinasikan jenis *low impact* dan *high impact*. Gerakan ini dimaksudkan untuk emmberi variasi agar tidak jenuh dan cepat lelah.

2.6 Pengertian Nutrition

Menurut Sharon Stoll, Ph.D (1986:36) *nutrition* (nutrisi) adalah proses dimana tubuh manusia menggunakan makanan untuk membentuk energi, mempertahankan kesehatan dan pertumbuhan untuk berlangsungnya fungsi normal setiap organ baik antara asupan nutrisi dengan kebutuhan nutrisi. Untuk orang yang latihan/berolahraga, diet yang optimal berfungsi untuk memasok nutrisi yang memadai sebagai pemeliharaan jaringan, perbaikan, dan pertumbuhan.

Tujuh kelompok makanan yang baik untuk dikonsumsi adalah lauk pauk hewan dan nabati sebagai penyedia protein, sayuran hijau sebagai penyedia vitamin dan mineral, sayuran kuning sebagai penyedia serat dan karbohidrat, buah-buahan sebagai penyedia vitamin, roti/sereal sebagai penyedia karbohidrat, susu sebagai penyedia kalsium dan mineral, dan linoleic acid yang terdapat pada mentega dan minyak zaitun. Seseorang dapat mengkonsumsi lebih dari satu kali asupan vitamin yang cukup untuk memenuhi kebutuhan sehari-hari. Karena vitamin dapat digunakan berulang kali dalam proses metabolisme. Vitamin untuk kebutuhan seorang atlet dan orang-orang yang aktif latihan/berolahraga tidak lebih besar dari yang bukan atlet.

Sebagai individu yang lebih aktif, kebutuhan untuk konsumsi energi meningkat juga meningkat, yang menjelaskan mengapa seorang pelari maraton

dapat mengkonsumsi sebanyak 5000 kalori per hari. Berikut adalah asupan nutrisi yang direkomendasikan untuk orang dewasa yang aktif:

1. Protein

   Rekomendasi standar untuk asupan protein 0,8 gram protein per kg berat badan. satu kg sama menjadi 2,2 pounds, sehingga seseorang dengan berat 130 pon membutuhkan 47,2 gr protein. Seseorang 170-lbs akan membutuhkan total rata-rata 62 gram kalori. Akan tetapi kita sebaiknya tidak mengkonsumsi protein lebih dari dua kali atau terlalu banyak, karena surplus protein tidak meningkatkan massa otot kalori tambahan melainkan diubah menjadi lemak dan dapat menambah ketegangan pencernaan pada hati dan ginjal.

2. Fats

   Asupan *fats* atau lemak yang optimal adalah tidak lebih dari 20 persen dari jumlah kalori total yang kita konsumsi. Tentu saja, seseorang dengan penyakit kandung empedu dan penyakit jantung harus menghindari kelebihan tingkat lemak. Sedangkan terlalu rendah konsumsi lemak dapat mengurangi penyerapan vitamin oleh tubuh karena vitamin larut lemak harus masuk ke dalam tubuh melalui konsumsi lemak esensial yaitu asam lemak dan asam linoleat, sehingga diet bebas lemak sangat tidak dianjurkan dan berpotensi berbahaya.

3. Carbohidrate

   Asupan *carbohidrate* atau karbohidrat harus setidaknya 40 sampai 50 persen dari total kalori individu. Makanan kaya karbohidrat seperti biji-bijian, tepung, kacang-kacangan sangat bermanfaat bagi mereka yang memakannya.

Makanan dan pola konsumsi dapat memberikan informasi tentang mengapa orang yang kelebihan berat badan biasanya terjadi karena terlalu banyak dan terlalu sering makan, tanpa disertai latihan yang memadai. Studi menunjukkan bahwa orang yang berolahraga secara teratur memiliki kontrol terhadap nafsu makan. Mereka dapat mengatur persis berapa banyak makanan yang harus dikonsumsi.

Sebagian besar dari kita terlalu banyak makan gorengan, makanan berlemak. Hanya sebagian kecil dari kita makan Tujuh kelompok makanan yang baik untuk dikonsumsi setiap hari. Bahkan ada juga orang-orang yang melewatkan sarapan, mengkonsumsi makan siang dengan porsi sedikit, kemudian makan malam degan porsi banyak, diikuti oleh *snack* (cemilan) sampai dengan mereka tidur. Jika anda tidak sarapan pagi, anda akan kehabisan energi sebelum siang hari. Kenneth Copper memiliki teori bahwa semua kalori harus dikonsumsi sebelum pukul lima sore. Dia berpendapat bahwa semua kalori diproses oleh tubuh pada saat tubuh banyak beraktivitas mulai dari sarapan di pagi hari sampai dengan menjelang makan malam, karena setelah makan malam sebagian besar dari kita biasanya hanya membaca, menonton televisi, atau tidur.

2.7 Body Mass Index (BMI)

Body Mass Index (BMI) merupakan suatu pengukuran yang menunjukkan hubungan antara berat badan dan tinggi badan. BMI merupakan suatu rumus matematika dimana berat badan seseorang (dalam *kg*) dibagi dengan tinggi badan (dalam m^2). Rumus menghitung nilai BMI dapat dilihat pada gambar 2.1.

Gambar 2.1 Rumus Kalkulasi BMI

BMI lebih berhubungan dengan lemak tubuh dibandingkan dengan indikator lainnya untuk tinggi badan dan berat badan. Seseorang dengan BMI 25-29,9 dikatakan mengalami kelebihan berat badan (*overweight*), sedangkan seseorang dengan BMI 30 atau lebih dikatakan mengalami obesitas. BMI bisa memperkirakan lemak tubuh, tetapi tidak dapat diartikan sebagai persentase yang pasti dari lemak tubuh. Hubungan antara lemak dan BMI dipengaruhi oleh usia dan jenis kelamin. Wanita lebih mungkin memiliki persentase lemak tubuh yang lebih tinggi dibandingkan pria dengan nilai BMI yang sama.

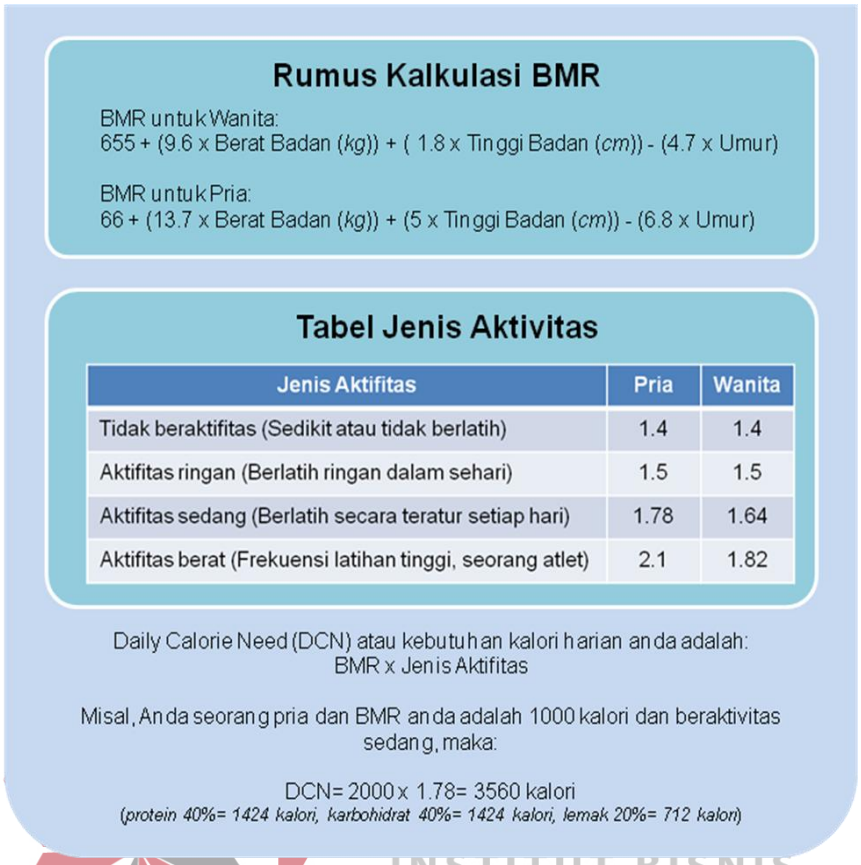
Pada BMI yang sama, orang yang lebih tua memiliki lebih banyak lemak tubuh dibandingkan orang yang lebih muda. BMI yang sehat untuk dewasa adalah 18,5-24,9. BMI yang tinggi merupakan suatu ramalan kematian karena penyakit jantung dan pembuluh darah. Diabetes, kanker, tekanan darah tinggi dan osteoartritis juga merupakan akibat dari kelebihan berat badan (*overweight*) dan obesitas yang sering ditemukan pada dewasa. Obesitas sendiri merupakan faktor resiko yang kuat dari kematian dini.

2.8 Basal Metabolic Rate (BMR)

Basal Metabolic Rate (BMR), dapat didefinisikan sebagai kebutuhan kalori individu yang dibutuhkan untuk metabolisme dalam tubuh pada saat seseorang dalam keadaan istirahat atau diam tetapi bukan sedang tidur. Dalam program penurunan berat badan, mengetahi BMR akan sangat membantu terutama dalam hal penatalaksanaan diet. Kebanyakan diet yang dilakukan hanya asal mengurangi asupan kalori tanpa mempertimbangkan BMR. Metabolisme merupakan kata yang terdengar akrab bagi sebagian orang yang sedang menjalankan program diet.

Orang yang gemuk akan diasumsikan sebagai orang dengan metabolisme lambat sehingga mudah menumpuk kalori dalam bentuk cadangan lemak, sebaliknya orang yang kurus dianggap memiliki metabolisme yang cepat sehingga makanan yang masuk langsung dimetabolisme untuk kebutuhan tubuh. Cepat atau lambatnya metabolisme kita dipengaruhi oleh faktor genetika.Namun demikian metabolisme juga dapat diatur sesuai dengan kebutuhan kita. Metabolisme dapat berubah-ubah bahkan kita dapat mengaturnya. Metabolisme inilah yang sering disebut sebagai BMR. Dengan bertambahnya usia secara otomatis juga akan menurunkan BMR.

BMR menjadi penting dan berguna setelah faktor aktifitas sehari-hari ikut diikutsertakan dalam kalkulasi dan kemudian menjadi Kebutuhan Kalori Harian atau *Daily Caloric Needs* (DCN). Bila Kebutuhan Kalori Harian sudah diketahui, maka akan menjadi lebih mudah menyusun diet, baik itu untuk *bulking*, *cutting*, maupun sekedar *maintenance* saja. Rumus menghitung nilai BMI dapat dilihat pada gambar 2.2.



Gambar 2.2 Rumus Kalkulasi BMR

Setelah Kebutuhan Kalori Harian diketahui, baik untuk bulking, cutting, atau maintenance, tinggal menerjemahkan jumlah kalori tersebut menjadi kuantitas makanan. Misalkan Kebutuhan Kalori Harian yang saya butuhkan 2000 kalori maka cara menerjemahkannya adalah sebagai berikut:

1. Tentukan komposisi *makronutrien* (karbohidrat:protein:lemak) yaitu 40:40:20. Karena total kalori adalah 2000 kalori maka komposisinya adalah:
   a. Karbohidrat = 800 kalori
   b. Protein = 800 kalori
   c. Fat = 400 kalori

2. Gunakan pedoman 1 *gram* karbohidrat sama dengan 4 kalori, 1 *gram* protein sama dengan 4 kalori, 1 *gram* fat sama dengan 9 kalori. Contohnya, dalam 200 *gram* protein berarti mengandung 800 kalori.
3. Tentukan kandungan *makronutrien* dalam bahan makanan. Dalam 100 *gram* dada ayam terdapat 25 *gram* protein, maka untuk mencapai kebutuhan 200 *gram* protein per hari dada ayam yg harus dikonsumsi adalah sebanyak 800 *gram*. Lakukan hal yang sama untuk karbohidrat dan fat.
4. Jika anda berencana makan sebanyak lima kali dalam sehari maka bagi menjadi lima kebutuhan harian masing-masing *makronutrien*. Misalnya, kebutuhan protein harian anda adalah 200 *gram* dan berencana makan lima kali sehari maka setiap kali anda makan harus mengkonsumsi 200 *gram* protein dibagi menjadi lima yaitu 40 *gram* protein atau setara dengan 160 *gram* daging dada ayam. Lakukan hal yang sama untuk karbohidrat dan fat.

2.9 Maximum Heart Rate (MHR)

Maximum Heart Rate (MHR) adalah jumlah tertinggi dari detak jantung dapat berdenyut dalam waktu satu menit, atau detak jantung yang dianjurkan pada seseorang yang melakukan aktivitas fisik maksimal. Denyut jantung ditentukan oleh jumlah detak jantung per satuan waktu, biasanya dinyatakan sebagai denyut per menit/*beat per minute* (bpm), dapat bervariasi dengan sebagai kebutuhan tubuh untuk perubahan oksigen, seperti selama latihan atau tidur. Pengukuran denyut jantung digunakan oleh profesional medis untuk membantu dalam diagnosis dan pelacakan kondisi medis. Hal ini juga digunakan oleh individu, seperti atlet, yang tertarik dalam memantau detak jantung mereka untuk

mendapatkan efisiensi maksimum dari pelatihan mereka. Rumus menghitung nilai BMI dapat dilihat pada gambar 2.3.

Gambar 2.3 Rumus Kalkulasi MHR

Jantung berdetak dewasa rata-rata sekitar 60 sampai 80 kali per menit saat istirahat. Tingkat jantung pada saat istirahat biasanya meningkat seiring dengan bertambahnya usia, dan itu umumnya menjadi lebih rendah pada orang yang sehat secara fisik. Denyut jantung istirahat digunakan untuk menentukan sasaran pelatihan denyut jantung seseorang. Seorang atlet umumnya mengukur laju jantung istirahat mereka sebagai salah satu cara untuk mengetahui jika mereka lebih terlatih. Denyut jantung beradaptasi terhadap perubahan kebutuhan tubuh akan oksigen, seperti selama latihan atau tidur.

2.10 Unified Modeling Language (UML)

Unified Modeling Language adalah salah satu *tools* yang paling penting dalam pengembangan sistem saat ini. UML memungkinkan pengembang sistem

untuk membuat *blueprint* yang menangkap visi mereka dalam sebuah standarisasi, mudah dimengerti, dan dapat mengkomunikasikan antar mereka dalam satu tim.

Mengkomunikasikan visi adalah sesuatu yang sangat penting. Sebelum melanjutkan UML, pengembang sistem sering mencoba menangkap usulan-usulan. Analis sistem mencoba menangkap keperluan dari klien mereka, men-*generate* analisis keperluan dalam beberapa notasi yang dimengerti analisis, memberikannya ke *programmer*, dan berharap hasil akhir seperti yang diinginkan oleh analisis.

Dikarenakan pengembangan sistem adalah juga manusia biasa, maka sangat berpontensial untuk mengalami kesalahan dalam setiap *stage* proses. Analisis mungkin mengalami kesalahpahaman dengan klien. Analisis mungkin menghasilkan dokumen-dokumen yang mana klien tidak menghendakinya. Hasil rancangan analisis mungkin tidak jelas buat *programmer*, sehingga *programmer* menghasilkan program yang sulit digunakan oleh klien dan bukan suatu solusi dari persoalan-persoalan dasar klien mereka.

Sejarah UML menurut Boogs (2002:24) UML adalah buah pikiran dari Grady Booch, James Rumbaugh, dan Ivar Jacobson. Mereka bekerja dalam organisasi yang terpisah antara tahun 80-an sampai dengan awal 90-an. Mereka merencanakan sebuah metodelogi untuk analis dan desain yang berorientasi obyek. Pada pertengahan 90-an mereka meminjam ide-ide dari yang lainnya, sehingga mereka merencanakan menyelesaikan pekerjaan mereka secara bersama-sama.

Pada tahun 1994 Rumbaugh bergabung dengan Rational Software Corporation, dimana Booch sudah bekerja disana. Jacobson mendaftarkan diri

pada tahun berikutnya. Mereka mengatakan versi *draft* UML dimulai dari industri *sofware* dan menghasilkan umpan balik perubahan secara substansial. Banyak perusahaan menyatakan bahwa UML akan melayani tujuan strategi mereka, sebuah konsorsium UML. Hewlett-Packard, Intellcorp, Microsoft, Oracle, Texas Instruments, Rational, dan yang lainnya. Konsorsium menghasilkan versi 1.0 UML dan mengajukannya ke *Object Management Group* (OMG) dan permohonan OMG untuk diajukan sebagai bahasa modeling standar.

Konsorsium dikembangkan menghasilkan versi 1.1 dan diajukan pada OMG, yang mana diadopsi akhir tahun 1997. Tahun 1998 OMG mempertahankan UML dan menghasilkan dua revisi. Akhirnya UML secara *de facto* menjadi standar dalam industri *software*.

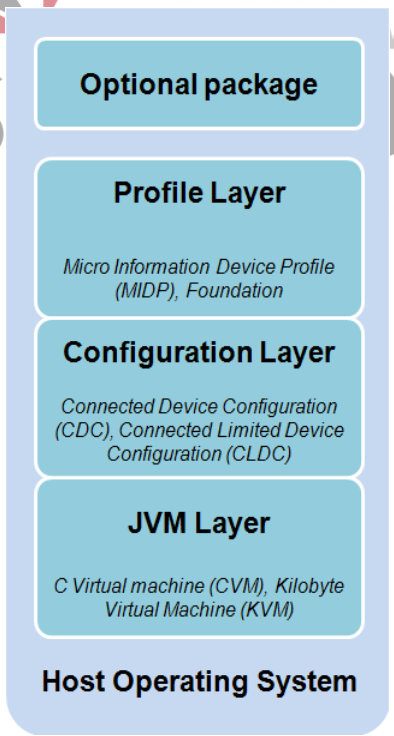
UML terdiri atas sejumlah elemen-elemen grafik yang mengkombinasikan ke dalam bentuk diagram dikarenakan ia adalah sebuah bahasa. UML mempunyai aturan untuk mengkombinasikan elemen-elemennya. Tujuan dari diagram-diagram ini adalah untuk menghasilkan *multiple view* dari sistem, dan kumpulan dari *view* disebut model. Model UML dari suatu sistem suatu saat seperti sebuah model skala dari bangunan. Penting untuk diperhatikan bahwa model UML menjelaskan apa yang diajukan sistem untuk dikerjakan, bukan bagaimana cara mengimplementasikannya.

2.11 Java 2 Micro Edition (J2ME)

Menurut Nyura (2010:19) *Java* 2 *Micro Edition* adalah lingkungan pengembangan yang didesain untuk meletakkan perangkat lunak Java pada barang elektronik beserta perangkat pendukungnya. Pada J2ME jika perangkat lunak

berfungsi dengan baik pada sebuah perangkat, maka belum tentu berfungsi baik pada perangkat yang lainnya. J2ME membawa Java ke dunia informasi, komunikasi dan perangkat komputasi selain perangkat komputer *desktop* yang biasanya lebih kecil dibandingkan perangkat komputer *desktop*. J2ME biasanya digunakan pada telepon seluler, *pager*, *Personal Digital Assistants* (PDA), dan sejenisnya (Shalahuddin dan Rossa, 2006).

J2ME merupakan sebuah teknologi pemrograman yang khusus dikembangkan oleh Sun Microsystem sebagai bahasa pemrograman untuk perangkat dengan memori yang sangat terbatas seperti telepon seluler (Rasmana, Petrus, Ari, 2005). Dengan menggunakan teknologi J2ME, aplikasi dapat diimplementasikan ke perangkat telepon seluler karena ukuran aplikasi yang relatif kecil. Arsitektur J2ME dapat dilihat pada gambar 2.4.



Gambar 2.4 Arsitektur J2ME

Perangkat *mobile* memiliki beragam jenis dan spesifikasi yang didesain untuk tujuan yang berbeda-beda. Bahkan dengan jenis yang sama, masing-masing perangkat memiliki spesifikasi berbeda sehingga perlu adanya suatu standar guna mengatasi masalah portabilitas aplikasi. Untuk menyeimbangkan antara portabilitas dengan performa, J2ME memiliki beberapa komponen yang lebih dikenal dengan C*onfiguration, Profile* dan *Optional Packages*. Kombinasi antara sebuah C*onfiguration* dengan sebuah *Profile* mendefinisikan perangkat tertentu. *Configuration* menyediakan fungsi-fungsi standar dan umum dari library J2ME. *Profile* berada di atas C*onfiguration* dan menyediakan *library* tambahan yang spesifik seperti konektifitas jaringan, keamanan dan tampilan *user interface*. *Optional Packages* dapat dimasukkan kedalam *Profile* untuk mendukung kebutuhan tertentu dari aplikasi (Sharp dan Yuan, 2004). Java juga dapat melakukan rekayasa file *image*/gambar seperti mengambil, menulis dan merubah ukuran, salah satunya adalah menggunakan *class* ImageIO. ImageIO merupakan class bantu yang berfungsi untuk melakukan input/output file bertipe image. Karena merupakan *class* input/output, maka ImageIO akan *"throws"* IOException jika terjadi suatu *error*.

Untuk mendukung berbagai jenis peralatan dari berbagai produsen, J2ME mengimplementasikan konsep *multi-layer*. J2ME menerapkan tiga lapisan (*layer*) perangkat lunak di atas sistem operasi dari peralatan yang bersangkutan. Ketiga lapisan tersebut adalah:

1. Lapisan KVM

   Lapisan JVM pada J2ME disebut C *Virtual Machine* (CVM) dan *Kilobyte Virtual Machine* (KVM). Lapisan ini merupakan implementasi dari JVM

yang telah disesuaikan dengan sistem operasi tertentu dan mendukung konfigurasi J2ME tertentu pula.

2. Lapisan Configuration

   Sebuah *Configuration* menentukan pustaka kelas untuk sekelompok peralatan komputasi berdasarkan kebutuhan memori dan kemampuan pemrosesan. J2ME memiliki dua konfigurasi, yaitu *Connected Device Configuration* (CDC) dan *Connected Limited Device Configuration* (CLDC).

3. Lapisan Profile

   Lapisan *Profile* berada di atas lapisan konfigurasi. Lapisan ini menentukan pustaka kelas untuk satu jenis peralatan elektronis tertentu. Sebuah perangkat lunak J2ME dibuat di komputer dengan mempergunakan bahasa pemrograman Java seperti halnya membuat perangkat lunak dengan komputer pada umumnya.

2.11.1 Connected Limited Device Configuration (CLDC)

Connected Limited Device Configuration adalah perangkat dasar dari J2ME, spesifikasi dasar yang berupa *library* dan API yang diimplementasikan pada J2ME, seperti yang digunakan pada telepon seluler, *pager* dan PDA. Perangkat tersebut dibatasi dengan keterbatasan memori, sumber daya dan kemampuan memproses. Spesifikasi CLDC pada J2ME adalah spesifikasi minimal dari *package*, kelas dan sebagian fungsi JVM (*Java Virtual Machine*) yang dikurangi agar dapat diimplementasikan dengan keterbatasan sumber daya pada alat-alat tersebut (Shalahuddin dan Rossa, 2006).

Peralatan–peralatan elektronik yang mempergunakan konfigurasi ini pada umumnya memiliki ciri-ciri sebagai berikut:

1. Koneksi ke jaringan tidak terlalu lama.
2. Memiliki antar muka yang sederhana.
3. Kapasitas memori antara 128 KB sampai 1 MB.
4. Mempergunakan prosessor 16 bit atau 32 bit.

Pada umumnya kelas-kelas yang ada pada konfigurasi ini diturunkan dari kelas-kelas yang ada di dalam J2SE. Kelas yang tidak diturunkan dari J2SE adalah kelas-kelas yang dipergunakan untuk melakukan koneksi ke jaringan.

2.11.2 Micro Information Device Profile (MIDP)

Micro Information Device Profile adalah profil dari J2ME yang dipergunakan untuk telepon seluler dan berada di atas konfigurasi CLDC. Pustaka kelas MIDP menyediakan API yang dapat dipergunakan untuk membuat dan memproses antar muka, basis data dan koneksi ke jaringan pada suatu perangkat lunak yang berjalan di dalam telepon seluler.

MIDP adalah spesifikasi untuk sebuah profil J2ME. MIDP memiliki lapisan di atas CLDC, API tambahan untuk daur hidup aplikasi, antarmuka, jaringan dan penyimpanan presisten. Pada MIDP versi 2.0 ke atas telah memiliki fitur untuk multimedia yang memiliki dukungan untuk memainkan *tone, tone sequence*, dan *file* WAV walaupun tanpa adanya Mobile Media API (Shalahuddin dan Rossa, 2006).

MIDP merupakan salah satu *profile* J2ME berbasis CLDC yang paling penting dan banyak digunakan pada perangkat telepon seluler seperti *smartphone*.

Spesifikasi MIDP dihasilkan dari suatu standar oleh *Java Community Process* (JCP) yang dibentuk dari sekumpulan perusahaan seperti Nokia, Motorola, Siemens, dan Symbian yang mendukung *profile* tersebut. Versi 1.0 dari MIDP memiliki standar *Application Program Interface* (API) yang telah ditentukan dan umumnya digunakan untuk aplikasi *mobile* seperti model aplikasi, *user interface*, penyimpanan tetap (*persistent storage*), jaringan, dan *timers*. MIDP versi 2.0 merupakan penyempurnaan dari versi sebelumnya dengan kemampuan kompatibilitas dengan versi 1.0 serta tambahan beberapa API seperti *games*, suara, *MIDlet signature* dan *Secure Connection* (HTTPS dan *Secure Socket*) (Riggs, 2003).

2.12 Record Management System

MIDlet tidak menggunakan file sistem untuk menyimpan data, tetapi menyimpan semua informasi dalam sebuah *memory non violate* (memori tidak tetap) yang disebut dengan *Record Management System* (RMS).

RMS (*Record Management System*) adalah kumpulan *record*, dan *record* disimpan sebagai *Array* dari *byte* dalam sebuah *record store* pada J2ME. RMS memiliki orientasi *record* basis data yang sederhana sehingga MIDlet dapat menyimpan informasi dan mengaksesnya.

2.13 Web Services

Web Services merupakan salah satu bentuk implementasi dari arsitektur *N-Tier*. Perbedaan *Web Services* dengan pendekatan *N-Tier* lainnya adalah dari segi infrastruktur dan dokumen yang digunakan sebagai format pertukaran data. Tentu saja *Web Services* sangat potensial bagi perkembangan kolaborasi aplikasi

Business to Business (B2B). Biasanya jika dua buah perusahaan ingin saling menukar informasi dapat dilakukan melalui proses yang panjang dan melelahkan. Sebagai contoh, mengatur sistem inventori perusahaan agar dapat berkomunikasi langsung dengan sistem produksi suplier pasti melalui proses negoisasi yang panjang, bagaimana memanggil suatu fungsi, seperti apa format dokumen yang akan ditukar, dan seterusnya. Dalam implementasinya *Web Services* tidak mempunyai tampilan, karena *Web Services* memang termasuk dalam *Tier Business Services*. Artinya didalam *Web Services* hanya tersedia fungsi-fungsi yang nantinya dapat digunakan oleh suatu aplikasi.

Selama lebih dari 20 tahun terakhir, *networking* dan *internet*, telah berevolusi secara signifikan. Pertama kali dikenal adanya suatu *transport* protokol yang merupakan jalur komunikasi untuk pertukaran data, contohnya adalah TCP/IP, dan di level yang lebih tinggi saat ini ada beberapa protokol aplikasi yang kita kenal seperti SMTP, FTP, dan *Gopher*. Setiap protokol mempunyai fungsi yang berbeda-beda. Sebagai contoh, FTP adalah suatu protokol yang memungkinkan kita untuk melakukan *file sharing* kepada orang lain. SMTP berfungsi untuk me-*routing* email dari si pengirim sampai ke alamat yang dituju. Kemudian muncul *HyperText Transport Protocol* (HTTP) dan *HyperText Document Language* (HTML). Melalui protokol dan format dokumen ini, memungkinkan kita untuk melakukan *sharing* dokumen yang kaya akan informasi kepada semua orang. User menggunakan aplikasi yang dinamakan browser untuk mengakses dokumen dengan format HTML. Hal ini membuat *internet* terbatas hanya digunakan untuk pertukaran informasi dengan manusia sebagai penggunanya.

Saat ini muncul *Extensible Markup Language* (XML) dan *Web Services*. *Web Services* didesain untuk mendayagunakan jaringan global yang ada saat ini yang dikenal dengan *internet*, dan juga termasuk *intranet*. XML merupakan suatu format dokumen yang berbasis teks. Dengan menggunakan format dokumen XML, *Web Services* memungkinkan suatu aplikasi berbicara dengan aplikasi lainnya. *Web Services* dapat di-implementasikan dalam berbagai platform menggunakan bahasa pemrograman apapun, dan bisa digunakan oleh berbagai platform menggunakan bahasa pemrograman apapun. Sebagai ilustrasi, suatu perusahaan A menggunakan sistem berbasis UNIX dan perusahaan B menggunakan sistem berbasis Windows. Pada masa yang lampau, transaksi diantara kedua perusahaan tersebut akan sulit untuk diimplementasikan. Dengan menggunakan *Web Services*, perusahaan A dapat menyediakan *interface* bagi perusahaan B melalui *Web Services* untuk melakukan transaksi data secara online. Konsep *Web Services* dapat dilihat pada gambar 2.5.

Gambar 2.5 Konsep Web Services

2.14 Hypertext Preprocessor (PHP)

Hypertext Preprocessor adalah *server side scripting envirotment* yang dapat digunakan untuk membuat dan menjalankan aplikasi-aplikasi di *web server* agar lebih interaktif dan *programmable.* Dengan adanya PHP, aplikasi-aplikasi yang ada di web server benar-benar dijalankan di *web server* tanpa mengharuskan adanya tambahan atau syarat tertentu untuk sisi klien (*web browser*). PHP biasanya dijadikan sebagai *module* dalam suatu web agar bisa mengeksekusi file-file PHP yang tersedia di *web server*. PHP dapat berjalan di hampir seluruh platform, *open source* dan berlisensi *GNU Public License* (GPL) (Welling, 2001).

PHP pada mulanya ditulis sebagai sebuah kumpulan dari CGI dengan menggunakan bahasa pemrograman C oleh *programmer* bernama Rasmus Lerdorf. *Programmer* asal Greenland ini membuat PHP pada tahun 1994 untuk menggantikan sebagian kecil kumpulan *script* dengan Perl yang digunakan untuk *maintenance* halaman web miliknya. Lerdorf mengawali menciptakan PHP untuk menampilkan *resume* miliknya dan mengumpulkan beberapa data, seperti berapa banyak lalu lintas data yang diterima dalam halaman web miliknya (Welling, 2001).

Setelah mengalami perkembangan oleh suatu kelompok *open source* (termasuk Rasmus), maka mulai versi 3 PHP memperlihatkan keunggulan sebagai salah satu bahasa *server* yang handal. Melalui perkembangan yang pesat ini banyak fasilitas yang ditambahkan oleh kelompok ini, maka jadilah PHP disebut sebagai *Hypertext Preprocessor*. Sintak yang digunakan berasal dari bahasa C, Java maupun Perl. Aplikasi yang dibangun dengan PHP memiliki kelebihan tersendiri. Beberapa kelebihan yang dimiliki PHP antara lain:

1. *Software* ini disebarkan dan dilisensikan sebagai perangkat lunak yang *open source*, maksudnya pendistribusian *master* programnya disertakan kode programnya dan biasanya secara gratis.
2. Dengan menggunakan PHP *script*, maka *maintenance* suatu situs web menjadi lebih mudah. Proses *update* data dapat dilakukan dengan menggunakan aplikasi yang dibuat dengan menggunakan *script* PHP.
3. Penulisan *script* PHP dapat menyatu dengan dokumen HTML, sehingga memudahkan pembuatannya. Untuk membedakan sintaks HTML dengan PHP, maka dibuatlah kesepakatan *tag* yang digunakan oleh PHP.
4. Kemampuan PHP yang paling diandalkan dan signifikan adalah dukungan kepada banyak *database*. Membuat halaman web yang menggunakan data dari *database* dapat sangat mudah untuk dilakukan. *Database* yang didukung oleh PHP antara lain: *adabas* D, *dBase*, *Empress*, IBM DB2, *Infomix*, *Ingers*, *Interbase*, *Frontbase*, *File Pro* (*read only*), *SQL Server*, *MySQL*, *Oracle*, ODBC, *PostgresSQL*, *Solid*, *Sysbase*, *Velocis*, dan unix DBM.

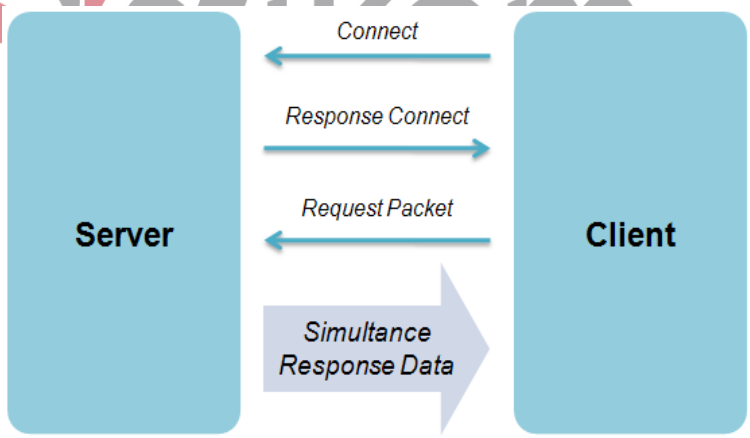
2.15 Socket

Definisi *Socket* adalah mekanisme komunikasi yang memungkinkan terjadinya pertukaran data antar program atau proses, baik dalam satu mesin maupun antar mesin. Menurut Makofske (2004:6) *Socket* dapat pula didefinisikan sebagai sebuah abstraksi, dimana suatu aplikasi dapat mengirim dan menerima sebuah data di jaringan komputer dalam waktu yang sama dan mengijinkan aplikasi membuka sebuah file untuk dibaca dan ditulis pada alat penyimpanan (*storage*). *Socket* mengijinkan sebuah aplikasi berjalan di dalam sebuah jaringan

dan berkomunikasi dengan aplikasi yang lain yang berjalan pada jaringan yang sama. Salah satu kelebihan dari komunikasi *Socket* yaitu mampu menangani banyak klien sekaligus (*multiple client*). Tipe utama *Socket* dibedakan menjadi dua bagian yaitu *Socket Stream* dan *Socket Datagram.* Jenis-jenis *Socket* tersebut dapat dijelaskan sebagai berikut:

1. Socket Stream

   Socket Stream menggunakan TCP sebagai *end to end* protokol (dengan IP dibawahnya). TCP *Socket* dalam melakukan komunikasi antara *node* satu dengan *node* yang lainnya. Salah satu *node* yaitu *client* harus melakukan koneksi langsung ke *node server* sehingga pada proses pengiriman *packet* akan dikirimkan secara simultan, yaitu melakukan pengiriman kembali jika *packet* yang telah dikirimkan hilang atau terjadi kegagalan (*lost packet*). Penjelasan lebih lanjut dapat dilihat pada gambar 2.6.

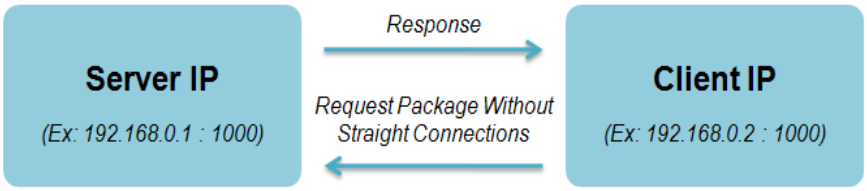


Gambar 2.6 Komunikasi Client-Server Dengan Socket Stream

2. Socket Datagram

   Socket Datagram menggunakan UDP sebagai *end to end* dengan IP dibawahnya, dimana data sampai ke *node* lain tanpa membuat koneksi dahulu

sehingga dalam perjalanan ke tujuan data dapat saja hilang karena tidak ada koneksi langsung antara kedua *host*. *Socket Datagram* merupakan jenis *Socket* yang tercepat dalam hal pengiriman sauatu paket atau data (*best-effort service*) yang mendukung pengiriman data dengan panjang sampai 65.500 byte. Penjelasan lebih lanjut dapat dilihat pada gambar 2.7.



Gambar 2.7 Komunikasi Client-Server Dengan Socket Datagram

2.16 MySQL

MySQL merupakan aplikasi *SQL Database Server* yang *multi-user* dan *ultithread* serta *open source* yang telah mempunyai lisensi *GNU Public Lisence* (GPL). Dengan kecepatan dan kemudahan dalam penggunaannya menyebabkan banyak aplikasi-aplikasi web yang berbasis database selalu menggunakan *MySQL* sebagai *database engine*-nya.

Versi terbaru *MySQL* telah dapat mendukung data spasial atau *Binary Large Object* (BLOB) tepatnya pada versi 4.1 ke atas. *MySQL* ini sendiri juga telah mendukung dan direkomendasikan oleh para pengembang *OpenGIS* (Wicaksono, 2005). Beberapa karakteristik yang dimiliki *MySQL* adalah :

1. *Application Program Interface* (API) untuk bahasa C, C++, *Eiffel*, *Java*, *Perl*, PHP, *Python*, dan TCL.
2. Bekerja pada banyak *platform* sistem operasi.

3. Mendukung penuh operator dan fungsi dalam sintaks SELECT dan WHERE sebagai bagian *query*.
4. Mampu mengolah *database* yang besar.
5. Mampu menerima data BLOB.

2.17 Interaksi Manusia dan Komputer

Menurut Rizky (2007:3), IMK (Interaksi Manusia dan Komputer) dideskripsikan sebagai sebuah disiplin ilmu yang mempelajari desain, evaluasi, implementasi dan sistem komputer interaktif untuk dipakai oleh manusia beserta studi tentang faktor-faktor utama dalam lingkungan interaksinya. Deskripsi IMK menurut Galitz (2002) dalam Rizky (2007:3) adalah suatu ilmu yang mempelajari perencanaan dan desain tentang cara manusia dan komputer saling bekerjasama sehingga manusia dapat merasa puas dengan cara yang paling efektif.

Menurut Rizky (2007:6), komponen-komponen penting dalam IMK yaitu interaksi, manusia, dan komputer. Interaksi adalah komunikasi yang terjadi antara manusia dan komputer. Jenis-jenis komunikasi tersebut antara lain C*ommand Entry*, *Menus and Navigation*, *Forms and Spreadsheets*, *Question and Answer Dialogue*, *Natural Llanguage Ddialogue*, *Windows Icon Menu Pointer* dan D*irect Manipulation.* Komponen selanjutnya yaitu manusia, yang dalam hal ini adalah pengguna yang dapat berupa seorang atau sekelompok pengguna, yang bekerja dalam sebuah tim atau organisasi dan saling berkaitan dalam mengerjakan tugas tertentu. Manusia dalam konteks IMK yang juga harus diperhatikan adalah komputer. Komputer diartikan sebagai perangkat keras ataupun perangkat lunak dari berbagai macam jenis yang nantinya berinteraksi dengan unsur manusia.

Galitz (2002) dalam Rizky (2007:26) menjelaskan bahwa sebelum memulai sebuah proses desain *interface*, terdapat beberapa tips desain yang harus diperhatikan, yaitu:

1. Memenuhi kaidah estetika

   Sebuah desain dapat disebut baik secara estetika jika (i) di dalamnya terdapat perbedaan yang jelas dan kontras antar elemen dalam sebuah tampilan. Misalnya tampilan tombol yang berbeda warna dengan tampilan textbox, (ii) terdiri dari beberapa kelompok yang jelas antara inputan dan tombol proses, (iii) antar elemen dan kelompok tampilan dipisah dengan jarak yang rapi, (iv) sederhana dan tidak terlalu banyak aksesoris yang terkesan sia-sia.

2. Dapat dimengerti

   Sebuah desain harus dapat dimengerti dengan cepat dari segi tampilan secara visual, fungsi yang akan ditonjolkan, penggunaan kata-kata yang singkat dan jelas, baik dalam tampilan maupun dalam perintah. Penggunaan metafora atau pemisalan yang berlebihan dalam sebuah fungsi harus dihindari.

3. Kompatibilitas

   Sebuah desain *interface* harus dapat memenuhi kompatibilitas dari berbagai segi antara lain (i) kompatibilitas pengguna yaitu dapat digunakan oleh pengguna dari kalangan yang lebih luas, baik berdasarkan strata pendidikan maupun berdasarkan usia, (ii) kompatibilitas penggunaan yaitu dapat memenuhi fungsi dan tujuan yang ingin dicapai dari perancangan sebuah perangkat lunak dan perangkat keras yang digunakan, (iii) kompatibilitas produk agar perangkat lunak dapat berjalan dengan baik di berbagai perangkat keras yang ada dan sistem operasi yang menjadi target aplikasi.

4. Komprehensif

   Sebuah sistem yang baik akan membimbing penggunanya agar dapat dan lebih mudah memahami apa yang harus diperhatikan, bagaimana cara melakukan sesuatu, kapan, dan dimana melakukan sesuatu serta mengapa harus melakukan sesuatu.

5. Konfigurabilitas

   Sebuah sistem harus dapat dikonfiguarasi ulang jika penggunanya menginginkan sesuatu berdasarkan fungsi tertentu.

6. Konsistensi

   Memiliki konsistensi dalam penempatan dan pemilihan gaya komponen visual, misalnya tombol atau ikon yang seragam.

7. Kontrol pengguna

   Pengguna dapat melakukan kontrol jika suatu saat terjadi kesalahan dalam proses serta pemilihan fungsi tambahan dari sebuah sistem. Hindari desain yang nantinya akan membatasi pengguna dalam memilih tampilan tertentu.

8. Efisien

   Desain dibuat se-efisien mungkin terutama dalam penempatan komponen, misalnya penenempatan tombol dalam sebuah panel yang dapat menarik perhatian pengguna.

9. Mudah dikenali

   Gunakan antar muka yang sudah dikenal oleh penggunanya, misalnya penempatan ikon *cut, copy,* dan *paste* secara standar dalam *toolbar*.

10. Toleransi

    Tidak ada sebuah sistem yang sempurna, dengan demikian terdapat beberapa toleransi kesalahan yang mungkin terjadi. Usahakan agar terjadi sebuah pesan yang dapat membimbing pengguna untuk keluar dari kesalahan yang terjadi.

11. Sederhana

    Lima cara untuk membuat desain sederhana dan tetap sesuai dengan keinginan pengguna yaitu (1) sembunyikan komponen visual jika tidak diperlukan, (2) sediakan pilihan standar, (3) minimalkan penggunaan berbagai macam *alignment*, (4) usahakan agar fungsi yang sering digunakan terlihat, dan (5) perhatikan konsep konsistensi.

2.18 Testing dan Implementasi Sistem

Menurut standart ANSI/IEEE 1059, *Testing* adalah proses menganalisa suatu entitas perangkat lunak untuk mendeteksi perbedaan antara kondisi yang ada dengan kondisi yang diinginkan (*defects/error/bugs*) dan mengevaluasi fitur-fitur dari entitas perangkat lunak.

Menurut Romeo (2003:3), *Testing Software* adalah proses mengoperasikan perangkat lunak dalam suatu kondisi yang dikendalikan untuk:

1. Verifikasi

   Apakah telah berlaku sebagaimana yang telah ditetapkan menurut spesifikasi?

2. Mendeteksi kegagalan (*error*)

3. Validasi

   Apakah spesifikasi yang ditetapkan telah memenuhi keinginan atau kebutuhan pengguna yang sebenarnya?

Menurut Romeo (2003:33), *Test Case* merupakan tes yang dilakukan berdasarkan pada suatu inisialisasi, masukan kondisi ataupun hasil yang telah ditentukan sebelumnya. Metode *Testing* ini dibagi menjadi dua, yaitu:

1. White Box Testing

   White Box Testing atau *Glass Box Testing* atau *Clear Box Testing* adalah suatu metode *Test Case* yang menggunakan struktur kendali dari desain prosedural. Metode desain *Test Case* ini dapat menjamin:

   a. Semua jalur (*path*) yang independen/terpisah dapat dites setidaknya satu kali tes.
   b. Semua logika keputusan dapat dites dengan jalur yang salah atau jalur yang benar.
   c. Semua *loop* dapat dites terhadap batasannya dan ikatan operasional.
   d. Semua struktur internal data dapat dites untuk memastikan validasinya.

2. Black Box Testing

   Black Box Testing atau *Behavioral Testing* atau S*pecification-based Testing*, input/output *Testing* atau F*unctional Testing* dilakukan tanpa sepengetahuan detail struktur internal dari sistem atau komponen yang dites. *Black Box Testing* berfokus pada kebutuhan fungsional pada perangkat lunak, berdasarkan spesifikasi kebutuhan dari perangkat lunak.

Menggunakan *Black Box Testing*, perekayasa perangkat lunak dapat menggunakan sekumpulan kondisi masukan yang dapat secara penuh memeriksa keseluruhan kebutuhan fungsional pada suatu program. Kategori *error* dapat diketahui melalui *Black Box Testing*, antara lain:

1. Fungsi yang hilang atau tidak benar.

2. *Error* dari antar muka.
3. *Error* dari struktur data atau akses eksternal *database*.
4. *Error* dari kinerja atau tingkah laku.
5. *Error* dari inisialisasi dan terminasi.