BAB 2
LANDASAN TEORI

2.1 Profil Tempat Penelitian

Berikut ini adalah profil CV. Cotelligent Indonesia Bandung sebagai skripsi dan penelitian dalam Optimasi Maintenance menggunakan Clean Code.

2.1.1 Sejarah Umum CV. Cotelligent Indonesia

CV. Cotelligent Indonesia adalah perusahaan yang bergerak di bidang jasa perekrutan pegawai dengan client yang tersebar di Eropa dan Kanada dan berkantor pusat di London. Perusahaan memiliki beberapa produk jasa yang digunakan untuk menjaring pasar dengan jenis bisnis yang sama maupun perusahaan non-outsourcing yang memiliki lowongan pekerjaan tersendiri.

2.1.2 Visi dan Misi CV. Cotelligent Indonesia

Berikut adalah visi dan misi yang dimiliki oleh perusahaan CV. Cotelligent Indonesia.

A. Visi Perusahaan

Visi perusahaan dapat dilihat sebagai berikut:

1. Menjaring pasar skala international karena reputasi yang baik.
2. Menjembatani kebutuhan perusahaan akan perekrutan pegawai dan kebutuhan masyarakat akan lowongan pekerjaan.

B. Misi Perusahaan

Misi perusahaan yang digunakan untuk mencapai visi, dapat dilihat sebagai berikut:

1. Membangkitkan kesadaran SDM akan determinasi dan komitmen yang tinggi terhadap pekerjaan.
2. Penggunaan SDM secara efisien dan bermoral tinggi.
3. Penggunaan sistem secara efektif.
4. Merespon dan memberikan layanan dengan baik.

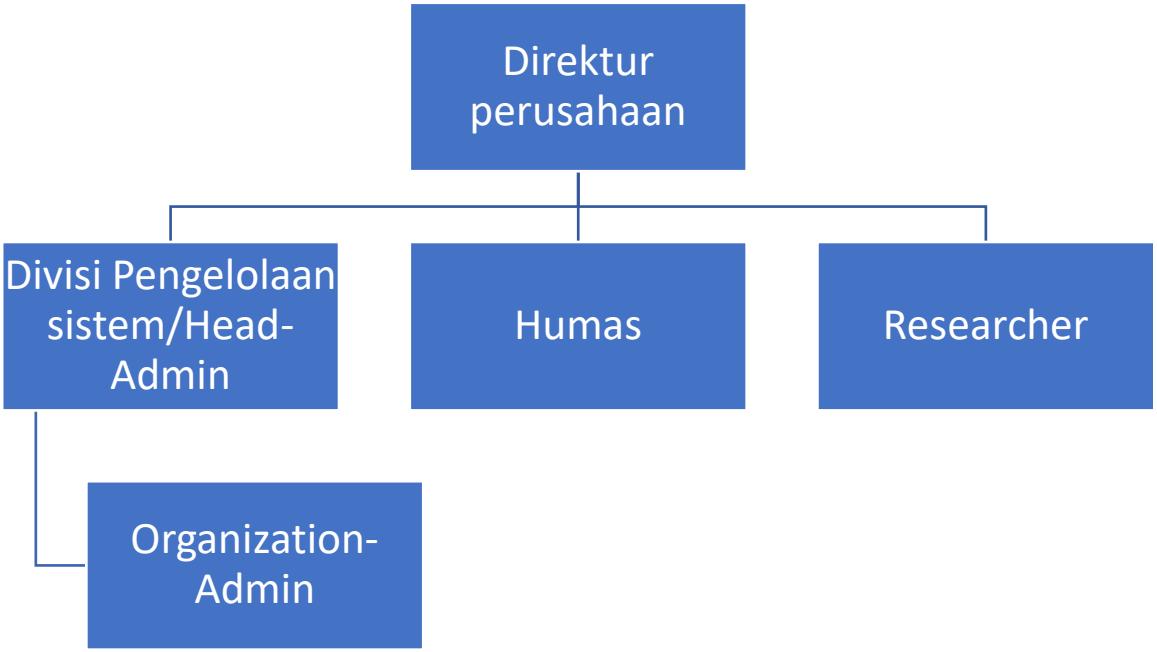
2.1.3 Logo dari CV. Cotelligent Indonesia

Berikut ini adalah logo dari perusahaan CV. Cotelligent Indonesia.

Gambar 2. 1 Logo Perusahaan

2.1.4 Struktur Organisasi

CV. Cotelligent Indonesia mempunyai struktur perusahaan yang terdiri dari beberapa divisi yang secara khusus tersusun dari berbagi bagian dan daerah pengoperasiannya. Struktur organisasi perusahaan memiliki peran yang penting untuk menjelaskan fungsi, tugas, tanggungjawab, dan wewenang perusahaan untuk mencapai mekanisme yang efektif dan efisien. Adapun struktur dari CV. Cotelligent Indonesia dapat dilihat pada gambar berikut :



Gambar 2. 2 Bagan Organigram Perusahaan

Deskripsi Jabatan dari Struktur Organisasi CV. Cotelligent Indonesia adalah sebagai berikut :

1. **Direktur Perusahaan** mempunyai tugas Memutuskan dan menentukan peraturan dan kebijakan tertinggi perusahaan, bertanggung jawab dalam memimpin dan menjalankan perusahaan, serta bertanggung jawab atas kerugian yang dihadapi perusahaan termasuk juga keuntungan perusahaan..
2. **Divisi Pengelolaan sistem/Head-Admin** menjadi peran penting dan mempunyai tugas untuk mengelola seluruh lapangan pekerjaan pada sistem yang berjalan, termasuk bertanggung jawab atas permasalahan yang terjadi pada sistem.
3. **Organization-Admin** memiliki tugas yang lebih spesifik yaitu untuk mengelola salah satu bidang pekerjaan saja.
4. **Humas** mempunyai tugas untuk mempromosikan kepada umum serta menjembatani klien dengan pekerjaan yang diinginkan.
5. **Researcher** memiliki tugas untuk melakukan seleksi pada kandidat tenaga kerja yang akan disalurkan pada perusahaan yang bekerja sama dengan CV. Cotelligent Indonesia agar tenaga kerja yang disalurkan terbukti kualitasnya dan siap kerja.

2.2 Landasan Teori

Untuk mendukung pembuatan laporan ini, maka perlu dikemukakan hal-hal atau teori-teori yang berkaitan dengan permasalahan dan ruang lingkup pembahasan sebagai landasan dalam pembuatan laporan ini. Landasan Teori ini mencakup Web, Database, PHP, MySQL, CSS, dan HTML.

2.2.1 Konsep Dasar Sistem

2.2.2 Pengertian Sistem

Sistem adalah suatu kesatuan utuh yang terdiri dari beberapa bagian yang saling berhubungan dan berinteraksi untuk mencapai tujuan tertentu.[1]

Sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari unsur atau variabel-variabel yang saling terorganisasi, saling berinteraksi, dan saling bergantung satu sama lain.[2]

.

2.2.2.1 Karakteristik Sistem

Model umum sebuah sistem adalah input, proses dan output. Hal ini merupakan konsep sederhana sebab sebuah sistem dapat memiliki beberapa masukan dan keluaran. Selain itu, sebuah sistem dapat memiliki karakteristik atau sifat-sifat tertentu mencirikan bahwa hal tersebut dapat dikatakan sebagai suatu sistem.[3] Adapun karakteristik yang dimaksud sebagai berikut :

1. Komponen Sistem (*Components*)

   Suatu Sistem terdiri dari beberapa komponen yang saling berinteraksi yang saling bekerja sama membentuk satu kesatuan. Komponen sistem tersebut dapat berupa suatu bentuk subsistem. Setiap Subsistem memiliki sifat dari sistem yang menjalankan suatu fungsi dan mempengaruhi proses sistem keseluruhan. Suatu sistem dapat mempunyai sistem yang lebih besar atau disebut juga “Super system”.
2. Lingkungan (*Environment*)

   Bentuk apapun yang ada diluar ruang lingkup atau batasan sistem yang mempengaruhi operasi sistem tersebut disebut lingkungan luar sistem. Lingkungan luar sistem dapat menguntungkan atau merugikan sistem tersebut. Dengan demikian, lingkungan luar tersebut harus tetap dijaga dan dipelihara. Lingkungan

luar yang merugikan harus dikendalikan. Jika tidak dilakukan, maka akan menggangu kelangsungan sistem tersebut.

3. Masukan (*Input*)

   Sumber daya atau energi yang dimasukkan ke dalam sistem disebut masukan, yang dapat berupa pemeliharaan dan sinyal.

4. Keluaran (*Output*)

   Hasil sumber daya atau energy yang diolah dan diklasifikasikan menjadi keluaran yang berguna. Keluaran ini merupakan masukan bagi subsistem yang lain. Keluaran yang dihasilkan adalah informasi yang dapat digunakan sebagai masukan untuk pengambilan keputusan atau hal-hal yang lain menjadi input bagi subsistem lain.

5. Pengolah (*Process*)

   Suatu sistem dapat mempunyai proses yang akan mengubah masukan menjadi keluaran.

6. Penghubung (*Interface*)

   Media yang menghubungkan sistem dengan subsistem lain. Memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem yang lain.

7. Batasan sistem (*Boundary*)

   Ruang lingkup sistem merupakan daerah yang membatasi antara sistem satu dengan sistem yang lain atau sistem dengan lingkungan luarnya.

8. Sasaran sistem (*Objective*)

   Suatu sistem memiliki tujuan dan sasaran yang pasti dan bersifat *deterministic*.

2.2.3 Pengertian Web

Menurut Arief (2011:8), Web adalah salah satu aplikasi yang berisikan dokumen-dokumen multimedia (teks, gambar, animasi, video) didalamnya yang menggunakan protokol HTTP (Hypertext Transfer Protocol) dan untuk mengaksesnya menggunakan perangkat lunak yang disebut browser.

2.2.4 PHP

PHP adalah Bahasa server-side –scripting yang menyatu dengan HTML untuk membuat halaman web yang dinamis. Karena PHP merupakan server-side-scripting maka sintaks dan perintah-perintah PHP akan diesksekusi diserver kemudian hasilnya akan dikirimkan ke browser dengan format HTML.[4]

Sedangkan menurut sumber lainnya, PHP atau singkatan dari Personal Home Page merupakan bahasa skrip yang tertanam dalam HTML untuk dieksekusi bersifat server side". PHP termasuk dalam open source product, sehingga source code PHP dapat diubah dan didistribusikan secara bebas. [5]

2.2.4.1 Kelebihan PHP

Seluruh aplikasi berbasis web dapat dibuat dengan PHP. Namun kekuatan yang paling utama adalah konektivitasnya dengan sistem database di dalam web. Kelebihan dari PHP adalah:

a. PHP mudah dibuat dan dijalankan, maksudnya PHP dapat berjalan dalam web server dan dalam sistem operasi yang berbeda pula.
b. PHP adalah *software open-source* yang gratis dan bebas didistribusikan kembali di bawah lisensi GPL (*GNU Public License*). User dapat mengunduh kode-kode PHP tanpa harus mengeluarkan uang atau khawatir dituntut oleh pencipta PHP.
c. PHP dapat dioperasikan pada platform Linux ataupun Windows.
d. PHP sangat efisien, karena PHP hanya memerlukan *resource system* yang sangat sedikit disbanding bahasa pemrograman lain.
e. Ada banyak Web server yang mendukung PHP, seperti Apache, PWS, IIS, dan lain-lain.
f. PHP juga didukung oleh banyak database, seperti MySQL, PostgreSQL, Interbase, SQL, dan lain-lain.
g. Bahasa pemrograman PHP sintaksnya sederhana, singkat dan mudah dipahami.
h. HTML-*embedded*, artinya PHP adalah bahhasa yang dapat ditulis dengan menempelkan pada sintak-sintak HTML.

2.2.5 HTML

HTML (*Hyper Text Markup Language)* adalah bahasa standar pemrograman untuk membuat halaman *web* yang terdiri dari kode kode *tag* tertentu, kemudian kode-kode tersebut diterjemahkan oleh *web browser* untuk menampilkan halaman *web* yang terdiri dari beberapa macam format tampilan seperti teks, grafik, animasi *link*, maupun audio-video.[6]

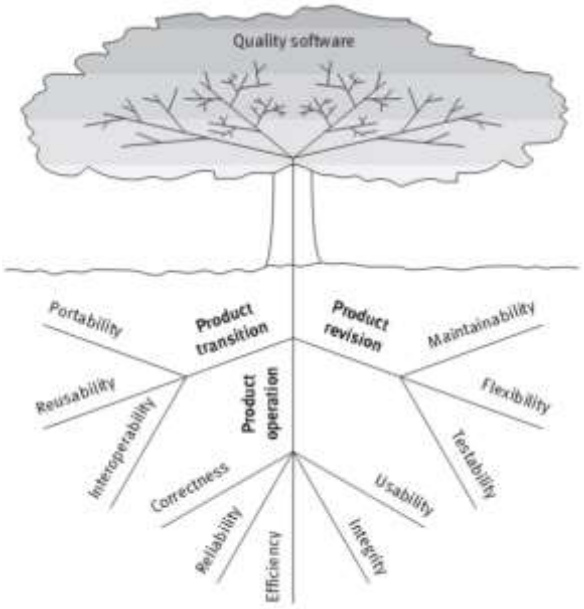
2.3 Software Quality Assurance (SQA)

Software Quality Assurance adalah suatu sistem dari sekumpulan tindakan yang bertahap dengan tujuan meyakinkan suatu produk sesuai dengan ketetapan persyaratan teknis.[7] Tujuan dari SQA merujuk pada beberapa aspek diantaranya aspek fungsional, fungsi manajerial dan fungsi ekonomi dari pemeliharaan dan pengembangan *software*.

Model pengujian *software* yang dapat digunakan yaitu model McCall, model pengujian McCall terdiri dari 11 faktor yang terbagi menjadi 3 kelompok [7,8], yaitu:

- Product Operation (Correctness,Reliability, Efficiency, Integrity, Usability).
- Product Revision (Maintability, Flexibility, Testability).
- Product Transisition (Portability, Reusability, Interoperability).

Berikut ini adalah gambaran pembagian faktor pengujian yang disebut Pohon McCall pada gambar.



Gambar 2. 3 Pohon McCall.[7]

Berdasarkan gambar diatas, setiap faktor memiliki sub faktor untuk memperjelas pengujian yang dapat dilihat pada tabel 2.1 ini.

Tabel 2. 1 Sub Faktor Dari Setiap Faktor Kualitas Software

Kategori	**Faktor Kualitas Software**	**Sub Faktor**
Product Operation	*Correctness*	*Accuracy*
		Completeness
		Up-to-dateness

		Availability
		Coding and Documentation Guidelines
		Compliance (Consistency)
	Reliability	*System Reliability*
		Application Reliability
		Computational Failur Recovery
		Hardware Failure Recovery
	Efficiency	*Efficiency Of Processing*
		Efficiency of Storage
		Efficiency of Communication
		Efficiency of Power Usage
Kategori	**Faktor Kualitas Software**	**Sub Faktor**
	Integrity	*Access Control*
		Access Audit
	Usability	*Operability*
		Training
Product Revision	*Maintability*	*Simplicity*
		Modularity
		Self-Descriptiveness
		Coding And Documentation Guidelines
		Compliance
		Document Accessbility
	Flexibility	*Modularity*
		Generality
		Simplicity
		Self-Descriptiveness
	Testability	*User Testability*
		Failure Maintenance Testability
		Tracibility
Product Transition	*Portability*	*Software System Independence*
		Modularity

		Self-Descriptive
	Reusability	*Modularity*
		Document Accessibility
		Software System Independence
		Application Independence
		Self Descriptive
		Generality
		Simplicity
	Interoperability	*Commonality*
		System Compatibility
		Modularity
		Software System Independence

2.4 Maintainability

Salah satu faktor yang terdapat pada SQA yaitu *Maintainability*, Fokus utama dari maintability yaitu pada kemudahan pemeliharaan pada software.[8] Maintainability menentukan upaya untuk melakukan pemeliharaan oleh tim untuk mengidentifikasi penyebab dari suatu kesalahan pada software, memperbaiki kesalahan yang ditemukan, dan melakukan pengujian terhadap koreksi kesalahan yang dilakukan.[7]

Kebutuhan umum *Maintability* antara lain:[7]

1. Pengembang mengikuti standar dan pedoman penulisan kode dari perusahaan pemilik *software*.
2. Ukuran modul *software* tidak lebih dari 30 pernyataan.

Menurut McCall, terdapat sub faktor dari beberapa faktor yang dapat dilihat pada tabel 2.2.

Tabel 2. 2 Sub Faktor Kualitas Software McCall

Faktor Kualitas Software	**Sub Faktor Kualitas Software**	**Keterangan**
Maintability	*Simplicity*	Kemudahan dalam memahami *Software*
	Modularity	Ukuran modul pada *Software*
	Self-Descriptiveness	*Source Code* yang mudah dipahami

	Coding and Document Guidelines	Ketersediaan petunjuk penulisan kode dan dokumentasi
	Compliance	Penggunaan desain dan teknik implementasi yang sejenis
	Document Accessibility	Kemudahan mengakses dokumen pada *software*

2.5 Maintainability Index (MI)

Maintainability Index (MI) merupakan suatu metrik komposit yang menggabungkan sejumlah metrik *traditional code* menjadi suatu nilai yang menyatakan nilai relatif dari aspek *maintainability*,[9][10] *Maintainability Index* terdiri atas metrik *Halstead Volume (HV)*, metrik *Cyclomatic Complexity (CC)*, rata-rata jumlah baris kode per modul (LOC). [12]

Adapun formula asli dari *Maintainability Index* dapat dilihat berikut ini:

$$MI = 171 - 5.2 * \ln(V) - 0.23 * (G) - 16.2 * \ln(LOC)$$

Nilai dari *Maintainability Index* dapat diukur dalam skala 0 hingga 100, yang mengartikan semakin tinggi nilai maintainability indexnya, maka semakin tinggi *maintainability* dari kode sumber yang diukur.[11]

Tabel 2. 3 Skala Penilaian Maintainability Index

Rentang	Keterangan
$20 \leq MI \leq 100$	Dapat dipelihara dengan baik
$10 \leq MI < 20$	Cukup untuk dapat dipelihara
$0 \leq MI < 10$	Sulit untuk dapat dipelihara

2.6 Halstead Complexity

Halstead Complexity merupakan salah satu metrik yang digunakan untuk mengukur kompleksitas dengan cara menghitung setiap operator dan operan yang terdapat

dalam *class*/modul pada suatu perangkat lunak.[13] Metrik ini biasa digunakan dalam metrik perhitungan *maintainability*. Pengukuran metrik ini terdiri atas:

1. *Operand* dan *Operator*
   Operan dan operator merupakan kunci dalam perhitungan *Halstead Complexity*, di mana simbol yang digunakan untuk menggambar kedua hal tersebut adalah sebagai berikut:

   n_1 = jumlah operator unik

   n_2 = jumlah operan unik

   N_1 = jumlah operator keseluruhan

   N_2 = jumlah operan keseluruhan

   Contoh Operator dalam perhitungan terdiri dari operator yang terdapat pada matematika, dan kata kunci dalam Bahasa pemrograman yang digunakan seperti if, for, dan var, symbol pemisah seperti tanda koma, titik koma, tanda kurung, sedangkan operan yang dimaksud yaitu variable, konstanta, angka, maupun kumpulan karakter.

2. Program Length
   Merupakan panjang dari program, yaitu hasil penjumlahan dari operator dan operan total.
   $$N = N_1 + N_2$$
3. Volume
   Volume adalah ukuran dari algoritma yang diterapkan pada program, perhitungan volume dilakukan dengan cara menghitung jumlah dari operator yang digunakan, dan operan yang terpakai pada algoritma, sehingga rumusnya seperti berikut:
   $$V = N * \log_2 n$$
4. Program Vocabulary
   Merupakan perhitungan yang dilakukan dengan cara menghitung hasil penjumlahan dari jumlah operator unik dan operan yang terdapat pada program, yang rumusnya dapat dilihat sebagai berikut:
   $$n = n_1 + n_2$$
5. Difficulty
   Metrik ini digunakan untuk melakukan pengukuran tingkat kerawanan terjadinya sebuah kesalahan pada program dengan menghitung proporsi jumlah operator unik yang ada pada program, dan proporsi rasio antara total jumlah operan dan jumlah operan unik pada program, yang rumusnya dapat dilihat sebagai berikut:
   $$D = (n_1/2) * (N_2/n_2)$$
6. Effort to Implement

Metrik yang digunakan untuk menghitung usaha memahami program yang dihitung berdasarkan nilai volume dan nilai difficuly dari program, berikut ini adalah rumusnya:

$$E = V * D$$

7. Time to Implement
   Metrik ini digunakan untuk mengukur waktu yang digunakan untuk memahami program dalam satuan detik, metric ini menghitung proporsi usaha atau dengan kata lain Effort to Implement, rumusnya dapat dilihat berikut:

$$T = E / 18$$

8. Number of Delivered Bugs
   Metrik yang digunakan untuk menghitung jumlah bug yang muncul pada program, metric ini memiliki korelasi dari keseluruhan program, berikut ini adalah rumusnya:

$$B = E_3^2 / 3000$$

2.7 Cyclomatic Complexity

Cyclomatic Complexity adalah metrik untuk mengukur tingkat kompleksitas yang terdapat pada sebuah fungsi dengan memperhatikan graf kendali, apabila fungsi tidak memiliki percabangan maka kompleksitasnya 1.[14] Jumlah *Cyclomatic Complexity* pada suatu fungsi disarankan kurang dari 15. Apabila melebihi 15, artinya terdapat sekitar 15 jalur eksekusi percabangan pada fungsi tersebut. Lebih dari itu, jalur eksekusi akan semakin sulit untuk diidentifikasi dan diperiksa. Adapun batas tertinggi jumlah jalur eksekusi dalam sebuah modul adalah 100. []

Jika fungsi memiliki banyak percabangan, maka untuk menghitungnya dapat menggunakan rumus berikut:

$$M = E – N + 2P$$

Keterangan:

M = Cyclomatic Complexity

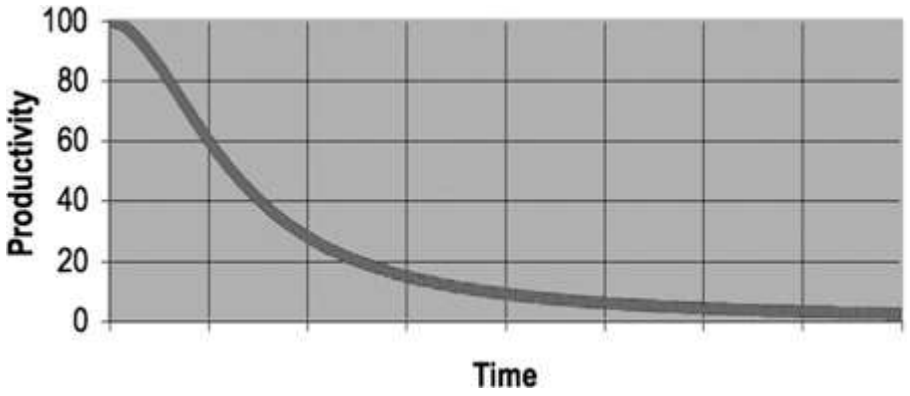
E = Jumlah edge pada paragraf

N = Jumlah node pada program

P = Jumlah Komponen yang terhubung

2.8 Clean Code

Clean code adalah suatu konsep penulisan struktur kode yang bersih, Yaitu kode yang mudah dibaca dan diubah oleh pengembang lain.[15] Clean code bertujuan untuk mengatasi penurunan tingkat produktivitas pengembangan perangkat lunak yang disebabkan struktur kode yang berantakan. Pada gambar dijelaskan bahwa waktu dapat mempengaruhi tingkat produktivitas.[15]



Gambar 2. 4 Productivity vs Time [15]

Berikut ini adalah hal pokok bahasan dalam clean code:

2.8.1 Meaningful Name

Merupakan konsep yang berisi petunjuk dalam pemberian nama variable, fungsi agar lebih mudah dipahami, berikut ini adalah petunjuk pemberian nama yang dapat digunakan:

1. Use Intention-Revealing Names
   Hindari pemberian nama suatu variable yang tidak memiliki maksud yang dapat membingungkan pengembang baru, contohnya adalah:
2. Avoid Disinformation
   Hindari pemberian kata yang maknanya berbeda dari makna yang sebenarnya.
3. Make Meaningful Names
   Memberikan perbedaan yang berarti dalam hal pemberian nama.
4. Use Pronounceable Names
   Gunakan nama yang dapat diucapkan dalam pemberian nama suatu variable atau fungsi.
5. Use Searchable Names
   Gunakan nama yang mudah untuk dicari dalam pemberian nama suatu variable atau fungsi.
6. Avoid Encodings
   Hindari pengkodean pada pemberian nama suatu variable atau fungsi.
7. Avoid Mental Mapping
   Hindari penggunaan singkatan untuk penamaan variable yang tidak diketahui umum dan mempersulit pemahaman dari kode yang ditulis.
8. Class Names
   Penamaan class disarankan menggunakan kata benda, namun hindari penggunaan kata kerja.
9. Method/Function Names
   Bertolak belakang dengan penamaan class, penamaan method disarankan menggunakan kata kerja.

2.8.2 Functions

Dalam menerapkan clean code pada fungsi, hal yang dapat dilakukan yaitu:

1. Do One Thing
   Fungsi yang baik hanya mengerjakan satu pekerjaan saja, agar fungsi dapat bekerja lebih cepat.
2. One Level of Abstraction per Function
   Untuk memastikan bahwa fungsi hanya mengerjakan satu pekerjaan saja, perlu memastikan bahwa fungsi tersebut berada dalam level abstraksi yang sama.
3. Reading Code from Top to Bottom
   Membuat fungsi yang dapat dibaca secara berurut dari atas hingga bawah, agar setiap fungsi mengikuti tingkat abstraksi selanjutnya sehingga program dapat dibaca, menurun satu tingkat abstraksi jika kita membacanya semakin ke bawah di dalam suatu fungsi, hal ini disebut juga *Step-down Rule*.
4. Use Descriptive Names
   Gunakan nama yang deskriptif dalam suatu fungsi.
5. Avoid *flag* as Parameter
   Hindari penggunaan *flag* sebagai parameter yang menjadikan fungsi dapat mengerjakan lebih dari satu pekerjaan.
6. Have No Side Effects
   Pastikan agar fungsi yang dibuat hanya memiliki satu pekerjaan saja, dan tidak memiliki efek samping terhadap hasil.
7. Don’t Repeat Yourself
   Jangan mengulangi diri sendiri (fungsi), Hal ini dapat menyebabkan duplikasi.
8. Error Codes Handling
   Merupakan petunjuk dalam melakukan penanganan kesalahan (Error Handling), agar dapat digunakan secara efisien dan pesan kesalahan mudah untuk ditampilkan, dengan tujuan menangani kesalahan dengan bersih dan tidak mengabaikan kesalahan yang ditemukan.

2.8.3 Comments

Dalam menerapkan clean code pada comment / komentar, clean code bertujuan untuk komentar yang ditulis dapat mengirimkan informasi yang tidak tersampaikan oleh source code yang ditulis.

1. Comments Do not Make Up for Bad Code
   Sebuah comments yang ditulis tidak seharusnya untuk menutupi sebuah Bad Code.
2. Explain Yourself in Code
   Sebuah comments yang baik dapat menjelaskan dirinya sendiri.
3. Good Comments

Sebuah comments yang diberikan memang diperlukan dan bermamfaat dalam penerimaan informasi. Beberapa poin dari good comments antara lain:

- Legal Comments
  Yaitu sebuah baris comment yang menunjukan legalitas dari sebuah *source code*, umumnya comments ini terdapat pada paragraph awal sebuah *source code*.
- Informative Comments
  Yaitu comments yang memberikan informasi dasar mengenai sebuah potongan *source code* agar mudah dipahami.
- Explanation of Intent
  Yaitu comments yang menjelaskan maksud dan niat dari potongan souce code, misalnya sebuah fungsi yang diberikan comments untuk menjelaskan mengerjakan apakah fungsi tersebut.
- Warning of Consequences
  Sebuah comments dapat menjelaskan kepada pengembang lain mengenai konsekuensi terhadap suatu kode jika dieksekusi

- Clarification
  Sebuah comments yang membantu menerjemahkan arti dari suatu argument atau pengembalian yang tidak jelas nilainya menjadi dapat terbaca pengembang baru.

4. Bad Comments
   Berbanding terbalik dengan good comments, bad comments berisi comments yang tidak jelas dan sulit untuk diterima sebagai sebuah informasi. Beberapa poin dari sebuah Bad Comments antara lain:
   - Mumbling
     Yaitu memberikan komentar karena pengembang merasa harus untuk melakukannya, namun komentar yang diberikan tidak membantu.
   - Redundant Comments
     Yaitu sebuah komentar yang mubazir misalnya komentar ditulis berulang-ulang.
   - Misleading Comments
     Yaitu sebuah komentar yang informasinya sesat, dan justru membingungkan.
   - Mandated Comments
     Yaitu sebuah komentar yang berisi pesan atau aturan yang justru mengacaukan, dan membingungkan.
   - Journal Comments

Yaitu sebuah komentar yang ditambahkan berisi log mengenai beberapa perubahan yang terjadi dari awal pembuatan hingga saat ini yang membuat komentar menjadi terlalu panjang.

- Noise Comments
  Yaitu komentar yang dinilai mengganggu bahkan dapat diabaikan karena tidak penting.
- Closing Brace Comments
  Yaitu komentar yang berisi komentar khusus dalam kurung kurawal,

2.9 Refactoring

Metode refactoring adalah metode menulis ulang sebuah kode program dengan cara memperbaiki sumber kode program tanpa mengubah *behaviour* sumber kode program sebelumnya, hal ini ditujukan untuk meningkatkan struktur dari kode program itu sendiri agar lebih baik. Pada penelitian ini code refactoring digunakan untuk membuat website sesuai dengan standar. Website standar dimaksudkan untuk menjadi dasar umum pada sebuah website sehingga browser dan perangkat lunak lain dapat memahami dengan mudah sebuah kode program diterjemahkan. Standar kode program yang digunakan

pada penelitian ini menggunakan standar pengkodean yang terdapat pada *codeigniter*.

Langkah untuk melakukan refactoring adalah sebagai berikut :[17]

1. Melakukan Tes Terhadap Kode
   Tes yang dilakukan terhadap kode dilakukan untuk mencari tahu letak kesalahan sumber kode program. Tes harus dilakukan sebaik mungkin sehingga dapat mengetahui dengan baik kesalahan yang terdapat pada sumber kode program tersebut. Menguji validasi standar kode program bisa menjadi salah satu tes yang bisa digunakan untuk mengetahui kesalahan yang terdapat pada sumber kode program.
2. Melakukan Perbaikan Sumber Kode Program
   Setelah menemukan kesalahan pada sumber kode program, sumber kode program tersebut diperbaiki. Perbaikan dilakukan hingga ketika dilakukan tes kembali terhadap sumber kode

program tidak menemukan kesalahan pada sumber kode program.

2.10 CodeIgniter

CodeIgniter adalah sebuah *framework* PHP yang dapat membantu mempercepat developer dalam pengembangan aplikasi web berbasis PHP dibanding jika menulis semua kode program dari awal.[16]

CodeIgniter pertama kali dibuat oleh Rick Ellis, CEO Ellislab, Inc. (http://ellislab.com), sebuah perusahaan yang memproduksi CMS (*Content Management System)* yang cukup handal, yaitu *Expression Engine.*

Saat ini, CodeIgniter dikembangkan dan dimaintain oleh *Expression Engine Development Team.*

Adapun beberapa keuntungan menggunakan CodeIgniter, diantaranya:

1. Gratis

   CodeIgniter berlisensi dibawah Apache/BSD opensorce.

2. Ditulis Menggunakan PHP 4

   Meskipun CodeIgniter dapat berjalan di PHP 5, namun sampai saat ini kode program CodeIgniter masih dibuat dengan menggunakan PHP 4.

3. Berukuran Kecil

   Ukuran CodeIgniter yang kecil merupakan keunggulan tersendiri. Dibanding dengan *framework* lain yang berukuran besar.

4. Menggunakan Konsep MVC

   CodeIgniter menggunakan konsep MVC yang memungkinkan pemisahan *layer application-logic* dan *presentation.*

5. URL yang Sederhana

Secara default, URL yang dihasilkan CodeIgniter sangat bersih dan *Serach Engine Friendly* (SEF).

6. Memiliki *Library* yang Lengkap

   CodeIgniter mempunyai *library* yang lengkap untuk mengerjakan operasi-operasi yang umum dibutuhkan oleh sebuah aplikasi berbasis web, misalnya mengakses *database*, mengirim email, memvalidasi form, menangani *session* dan sebagainya.

7. *Extensible*

   Sistem dapat dikembangkan dengan mudah menggunakan *plugin* dan *helper*, atau dengan menggunakan *hooks.*

8. Tidak Memerlukan *Template Engine*

   Meskipun CodeIgniter dilengkapi dengan *template* parser sederhana yang dapat digunakan, tetapi hal ini tidak mengharuskan kita untuk menggunakannya.

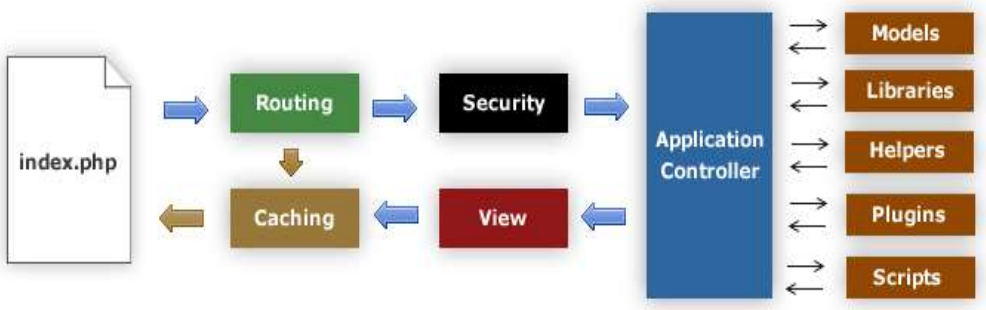
9. Dokumentasi Lengkap dan Jelas

   Dari sekian banyak *framework*, CodeIgniter adalah satu-satunya *framework* dengan dokumentasi yang lengkap dan jelas.

10. Komunitas

    Komunitas CodeIgniter saat ini berkembang pesat. Salah satu komunitasnya bisa dilihat di (http://codeigniter.com/forum/).

Proses aliran data aplikasi pada sistem dapat diilustrasikan seperti terlihat pada gambar :



Gambar 2. 5 *Application Flowchart*

Sumber : Hakim (2010 : 12) Membangun Web Berbasis PHP dengan *Framework* CodeIgniter

Keterangan :

1. Index.php berfungsi sebagai *front controller,* menginisialisasi *base resorce* untuk menjalankan CodeIgniter.
2. Router memerikasa HTTP *request* untuk menentukan apa yang harus dilakukan dengannya.
3. Jika *Cache* aktif, maka hasilnya akan langsung dikirimkan ke *browser* dengan mengabaikan aliran data normal.
4. *Security*. Sebelum *Controller* dimuat, HTTP *request* dan data yang dikirimkan *user* akan difilter untuk keamanan.
5. *Controller* memuat *model, core libraries, plugins, helpers* dan semua *resource* yang diperlukan untuk memproses *request.*
6. Akhirnya *View* yang dihasilkan akan dikirimkan ke *browser*. Jika *Cache* aktif, maka *View* akan disimpan sebagai *Cache* dahulu, sehingga pada *request* berikutnya langsung dapat ditampilkan.

2.11 Konsep MVC (Menu-View-Controller)

CodeIgniter adalah *framework* PHP yang dibuat berdasarkan kaidah *model-View-controller.* Dengan MVC, maka memungkinkan pemisahan antara *layer application-logic* dan *presentation*. Sehingga, dalam sebuah pengembangan web, seorang *programmer* bisa berkonsentrasi pada *core-system,* sedangkan web *designer* bisa berkonsentrasi pada tampilan web. Menariknya, skrip PHP, *query* MySQL, Javascript dan CSS bisa saling terpisah, tidak dibuat dalam satu skrip berukuran besar yang membutuhkan *resource* besar pula untuk mengesekusinya.

Dalam konteks CodeIgniter dan aplikasi berbasis web, maka penerapan konsep MVC mengakibatkan kode program dapat dibagi menjadi tiga kategori, yaitu :

1. *Model*

   Kode program (berupa OOP *class*) yang digunakan untuk memanipulasi *database*.

2. *View*

   Berupa *template* html/xml atau php untuk menampilkan data pada *browser*

3. *Controller*

   Kode program (berupa OOP *class*) yang digunakan untuk mengontrol aliran aplikasi (sebagai pengontol *model* dan *View)*