Rangkuman

pre BANGKIT Dicoding

Mulai Dasar Pemograman

Setiap orang harus belajar cara memprogram komputer, karena itu akan mengajarkan bagaimana cara berpikir.

-Steve Jobs-

 Yang terpenting adalah Anda harus bisa belajar mandiri, berkomitmen, benar-benar punya rasa ingin tahu

# BAB 1 - “Apa saja yang harus saya pelajari jika masih belum mengenal dunia pemrograman?”

Diharapkan **mampu memodifikasi HTML, CSS, dan JavaScript** tingkat dasar secara tepat sesuai persyaratan spesifikasi dan fungsionalitas aplikasi.

# BAB 2 – Kebutuhan Aplikasi Dari Segi Pengguna “ Docs, Mencari informasi client maunya apa ”

Diawali dari banyaknya cerita sejarah dimana sebuah roda tercipta dari kebutuhan orang. Dan sekarang kebutuhan makin banyak, dari berbisnis atau pun having fun. Tercipta lah website dll.

Proses Pengembangan aplikasi hingga perawatan (SDLC) :



Dalam membuat aplikasi pasti ada beragamnya kebutuhan pengguna, maka akan menjadi **mustahil untuk membuat solusi aplikasi yang bisa memenuhi kebutuhan semua pengguna secara sempurna.** Pasti akan bertemu dengan titik di mana kita harus mengambil keputusan dan akan muncul pertanyaan, **“Apakah keputusan ini menguntungkan? Bukankah ini akan merugikan kelompok tertentu?”**. Hal ini sudah menjadi lazim dalam pengembangan aplikasi, makanya banyak sekali opsi-opsi atau alternatif dalam dunia aplikasi, misalnya ada dua aplikasi yang memiliki tujuan sama akan tetapi fiturnya sangat berbeda.

## URS dibuat supaya stakeholder (pemegang kepentingan) dapat memahami suatu aplikasi yang ingin dibuat.

* **Cara gathering info untuk membuat sebuah aplikasi :**

1. Interview
   1. Tanya mereka taunya apa saja
   2. Liat gimana sebuah perusahaan bekerja
   3. Monitoring “Stalking” bagaimana orang2 disana bekerja
2. User Stories
   1. Sebagai seorang [deskripsi dari pengguna],   
      Saya menginginkan [suatu fungsionalitas],   
      sehingga mendapatkan [suatu kemampuan atau fitur tertentu].
3. Straw Man Dokumen
   1. storyboards, flowcharts, dan mock-up html.
4. Prototyping’

* Sebagai Contoh :
  + Ada Studi kasus Requirement login dan logout, Kan pasti setiap aplikasi beda2 login dan logout nya. Untuk perusahaan kita login dan logout nya mungkin berbeda,   
    Maka dari itu :
    - Kita tanya ke pengguna. *"Apa yang sebenarnya diharapkan pengguna ketika melakukan proses login atau logout?"*
    - Di halaman login seharusnya pengguna bisa melakukan proses login dengan memasukkan email dan password.
    - Ketika pengguna melakukan proses logout, ia akan secara otomatis pindah ke halaman login.
  + Oke, kita telah berhasil mendapat 2 user requirements.

## Spesifikasi Teknis Aplikasi

Beda dengan URS yang bahasanya ngobrol banget. Kali ini harus banyak bahasa teknisnya.

* Berikut ini adalah contoh spesifikasi teknis aplikasi untuk halaman checkout dari proses pembelian suatu barang:
  + *“Jika melakukan pembelian di atas 50 ribu rupiah maka otomatis tambahkan kupon diskon yang saat itu statusnya tersedia pada sistem, jika tidak ada kupon yang statusnya tersedia maka tampilkan tulisan tidak ada diskon yang tersedia”.*
  + Jika beli di atas 50K, langsung masukin kupon. Tapi kalo di akun gak ada kupon maka tampilkan tidak ada kupon.

dalam dunia pengembangan software **tidak ada ide/solusi yang sempurna**. Oleh karena itu, yang bisa dilakukan sebagai pengembang software adalah **memilih ide yang dirasa paling cocok** dan kemudian melakukan pembaharuan terhadap ide tersebut secara berkala mengikuti kebutuhan pengguna.

Siapkan Backup Plan.

## Menganalisis Spesifikasi Teknis Aplikasi

Disini Cuma menjelaskan, setelah kita membuat sebuah aplikasi yang hanya isinya Login dan Logout. Dan requirement yang di inginkan user.  
  
kita mengetes aplikasinya apakah sudah sesuai.  
  
**Jadi, Cuma retest apakah produk sudah sesuai dengan keinginan pembeli ( pengguna )**

## Tips Sikap Kerja dalam Meneliti, Menganalisis, dan Mengevaluasi Kebutuhan Aplikasi

Intinya. Dapatkan komunikasi yang baik dengan client agar produk akhir nya baik dan sesuai dengan yang di inginkan oleh pembeli ( client ). Dan pahami juga bagaimana aplikasi ini nanti di pake dengan penggunanya ( Karyawannya jika di sebuah perusahaan)  
  
1. Jalani apa yang di pengenin Pak BOS  
2. Lihat anak buah Pak BOS kerjanya gimana

## Rangkuman

**Rangkuman Memahami Kebutuhan Aplikasi**

User Requirement Specification (URS) adalah dokumen yang menyimpan informasi mengenai kebutuhan pengguna yang dibuat sebelum proses pengembangan suatu software. Dokumen ini bertujuan agar ketika ada yang membaca dokumen ini maka akan mendapatkan pemahaman yang sama akan kebutuhan user pada aplikasi.

Dokumen User Requirement Specification tidak boleh menggunakan jargon teknis yang hanya dipahami oleh kelompok tertentu, melainkan harus menggunakan bahasa yang bisa dipahami oleh seluruh stakeholder (pemegang kepentingan), seperti manajer, developer, customer, dan user.

Beberapa teknik requirement gathering yang bisa kita gunakan adalah:

* Interview, proses wawancara yang berupa tanya jawab
* User Stories, menuliskan kebutuhan user sesuai dengan role dan keinginannya
* Straw Man, menyampaikan ide tanpa menggunakan kode
* Prototyping, membuat sampel program yang hanya memiliki fitur-fitur utamanya saja

Spesifikasi Teknis Aplikasi yakni dokumen yang menyimpan informasi detail mengenai fungsionalitas dari sistem/aplikasi, servis, dan juga limitasi-limitasinya.

Beberapa prinsip yang perlu diperhatikan ketika membuat spesifikasi teknis aplikasi seperti berikut:

* Clear (jelas)
* Unambiguous (tidak ambigu)
* Mudah dipahami
* Complete (lengkap)
* Consistent (konsisten)

Dalam proses pembuatan spesifikasi teknis aplikasi kita perlu mempertimbangkan kebutuhan dari berbagai macam stakeholder. Berikut pandangan beberapa stakeholder terkait spesifikasi teknis:

* **Developer**  
  Dari sisi developer tentunya yang diperlukan ada kedetailan dan kejelasan spesifikasi. Karena merekalah yang akan mengimplementasinya ke dalam aplikasi.
* **Client/User**Dari sisi client meskipun spesifikasi teknis lebih mengarah ke teknis akan tetapi mereka berharap tetap bisa dimengerti. Dalam artian istilah teknis yang dipakai harus bisa dibuat semudah mungkin untuk dimengerti.
* **Legal**  
  Karena spesifikasi teknis ini bisa menjadi kontrak, maka perlu memasukkan acceptance criteria dengan jelas juga. Acceptance criteria adalah klausul kriteria yang berisi apakah suatu fitur sudah berjalan dengan baik. Jika aplikasi yang dibuat lulus semua acceptance criteria maka seharusnya tidak ada masalah dari sisi kontrak/legal.

Dalam mengembangkan aplikasi, sebaiknya disiapkan juga plan B atau contingency plan. Contingency plan adalah suatu tindakan alternatif yang dipersiapkan ketika tindakan utama yang direncanakan untuk melakukan sesuatu gagal atau terhambat oleh berbagai faktor.

# BAB 3 – Perencanaan Modifikasi Aplikasi “Docs, SRS dan Flowchart Stuff “

Membuat perencanaan modifikasi aplikasi perangkat lunak dengan membuat spesifikasi kebutuhan aplikasi perangkat lunak dan diagram alur (atau biasa disebut dengan flowchart).

## Software Requirement Specification (SRS)

Spesifikasi Kebutuhan Perangkat Lunak (SKPL) atau juga bisa disebut dengan Software Requirement Specification (SRS).

* Spesifikasi Kebutuhan Perangkat Lunak (SKPL) adalah sebuah dokumen yang dibuat sebelum mengembangkan sebuah aplikasi perangkat lunak.

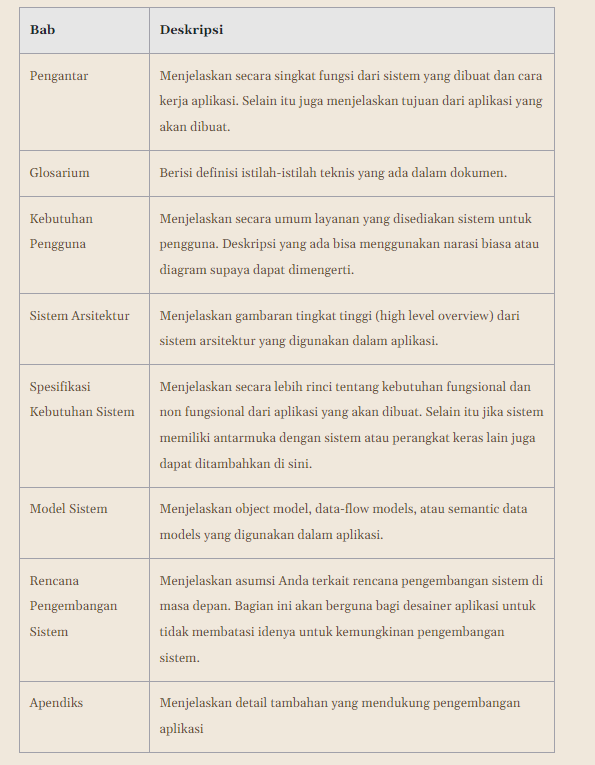
Dokumen ini menjelaskan **cara kerja** dan **kebutuhan fungsional** maupun **non-fungsional** dari aplikasi yang digunakan pengguna nantinya. Ini juga dapat menguntungkan developer agar tau nanti gambaran aplikasinya bagaimana.

Berikut cakupan elemen yang ada dalam dokumen Spesifikasi Kebutuhan Perangkat Lunak:

1. Mulai dari tujuan aplikasi;
2. deskripsi umum;
3. kebutuhan fungsional dan non fungsional;
4. performa aplikasi dalam proses produksi;
5. antarmuka eksternal atau bagaimana sebuah aplikasi berinteraksi dengan perangkat keras dan perangkat lunak lainnya;
6. hingga batasan sistem aplikasi yang akan dibuat.

Dari beberapa elemen tersebut mungkin Anda ingin bertanya, apa perbedaan dari kebutuhan fungsional dan non fungsional?

### Strukturnya



### Tahapan prosess Membuat SRS

Kali ini kita akan membuat sebuah dokumen SKPL untuk membuat autentikasi dari halaman website. Tenang saja, kita hanya membahas fitur **login dan logout dari halaman website secara sederhana**.

1. Pengantar
   1. Tujuan Aplikasi
   2. Sasaran Pengguna
2. Kebutuhan Pengguna
   1. Deskripsi umum dari aplikasi
   2. Kegunaan aplikasi bagi pengguna
3. Spesifikasi Kebutuhan Sistem
   1. Kebutuhan fungsional
   2. Kebutuhan non fungsional
   3. Antarmuka Pengguna (User interface)
   4. Antarmuka perangkat keras (Hardware interface)
4. Rencana Pengembangan Sistem

Dari kerangka tersebut, jabarkan setiap poin-poin yang ada mulai dari pengantar, hingga kesimpulan. Mari kita mulai dari tujuan aplikasi dan sasaran pengguna yang ada di bab pengantar. Anda dapat menuliskan bagian pengantar sebagai berikut:

|  |
| --- |
| Pengantar  **Tujuan Aplikasi:** Pengguna dapat mengakses suatu halaman web setelah melakukan proses autentikasi dengan memasukkan username dan password.  **Sasaran Pengguna:** Semua pengguna atau masyarakat umum yang sudah memiliki hak akses dalam website. |

Setelah itu, lanjutkan untuk menjabarkan kebutuhan pengguna yang berupa deskripsi umum dan kegunaan dari aplikasi untuk pengguna seperti berikut:

|  |
| --- |
| Kebutuhan Pengguna  **Deskripsi Umum Aplikasi:** Autentikasi dari sebuah halaman website adalah hal yang penting. Kita harus mengetahui elemen apa saja yang dibutuhkan untuk masuk ke halaman website seperti **email, password, dan tombol login**.  Ketika pengguna berhasil melakukan autentikasi **dengan memasukkan email dan password dengan benar, maka ia dapat mengakses konten website** yang ada di dalamnya. Sedangkan jika pengguna **tidak berhasil melakukan proses autentikasi, pengguna tidak dapat mengakses konten** yang ada dan harus memeriksa kembali apakah email dan password yang dimasukkan sudah benar.  Supaya lebih aman, ketika **pengguna selesai mengakses konten website, ia dapat melakukan proses logout** dengan menekan tombol logout yang ada di dalam halaman website tersebut. **Setelah logout, pengguna akan keluar dari konten halaman website dan harus melakukan autentikasi kembali** untuk bisa masuk.  **Kegunaan Aplikasi:**   * Pengguna dapat masuk (login) untuk mengakses konten yang ada di halaman website. * Aplikasi dapat memeriksa penulisan format email di dalam form pengisian email. * Pengguna dapat memasukkan password dengan aman karena antarmuka form password tersamarkan tampilannya. * Pengguna dapat keluar (logout) dari halaman konten website dengan menekan tombol logout. |

Kemudian jabarkan elemen-elemen yang ada di bab spesifikasi, seperti kebutuhan fungsional,non fungsional, antarmuka pengguna (***user interface***), antarmuka perangkat keras (***hardware interface***).

|  |
| --- |
| Spesifikasi Kebutuhan Sistem  **Kebutuhan Fungsional**   * **Pengguna dapat memasukkan email dan password pada form yang disediakan**. Kemudian ketika tombol login ditekan, pengguna yang sudah melengkapi email dan password untuk mengakses konten halaman website. * Pengguna dapat keluar dari konten halaman website dengan menekan tombol logout.   Kebutuhan Non Fungsional   * Ketika pengguna memasukkan email di dalam form email, sistem dapat **memeriksa apakah inputan tersebut sudah sesuai dengan struktur email yang benar**. Jika tidak sesuai, pengguna diberi pengingat bahwa belum mengisi form email dengan benar. * Ketika pengguna memasukkan password di dalam form **password,** sistem dapat **menyembunyikan setiap huruf yang diketikkan**. Sehingga, pengguna akan merasa aman ketika mengetik passwordnya di tempat umum sekalipun.   Antarmuka Pengguna (User interface)   * Perangkat lunak front-end: HTML dan CSS * Perangkat lunak back-end: JavaScript   Antarmuka perangkat keras (Hardware interface)   * Komputer atau ponsel cerdas dengan browser yang sudah terinstal di dalamnya |

Terakhir, uraikan rencana pengembangan sistem sistem, seperti berikut:

|  |
| --- |
| Rencana Pengembangan Sistem:  Dalam pengembangan sistem **ke depannya**, pengembang software **akan menambahkan fitur sebagai berikut**:   * Enkripsi password pada aplikasi autentikasi halaman website. * Pengecekan panjang password yang harus memiliki minimal 6 karakter. |

Nah, dari semua bab yang sudah kita buat, gabungkan menjadi satu dokumen SKPL Autentikasi Halaman Website. Sehingga **hasil akhirnya** seperti berikut:

|  |
| --- |
| Dokumen Spesifikasi Kebutuhan Perangkat Lunak Autentikasi Halaman Website  Pengantar   1. Tujuan Aplikasi: Pengguna dapat mengakses isi dari halaman web setelah melakukan proses autentikasi dengan memasukkan username dan password. 2. Sasaran Pengguna: Semua pengguna atau masyarakat umum yang sudah memiliki hak akses dalam website   Kebutuhan Pengguna   1. Deskripsi Umum Aplikasi: Autentikasi dari sebuah halaman website adalah hal yang penting. Kita harus mengetahui elemen apa saja yang dibutuhkan untuk masuk ke halaman website seperti email, password, dan tombol login.  Ketika pengguna berhasil melakukan autentikasi dengan memasukkan email dan password dengan benar, maka ia dapat mengakses konten website yang ada di dalamnya. Sedangkan jika pengguna tidak berhasil melakukan proses autentikasi, pengguna tidak dapat mengakses konten yang ada dan harus memeriksa kembali apakah email dan password yang dimasukkan sudah benar.  Supaya lebih aman, ketika pengguna selesai mengakses konten website, ia dapat melakukan proses logout dengan menekan tombol logout yang ada di dalam halaman website tersebut. Setelah logout, pengguna akan keluar dari konten halaman website dan harus melakukan autentikasi kembali untuk bisa masuk. 2. Kegunaan Aplikasi:    * Pengguna dapat masuk (login) untuk mengakses konten yang ada di halaman website.    * Aplikasi dapat memeriksa penulisan format email di dalam form pengisian email.    * Pengguna dapat memasukkan password dengan aman karena antarmuka form password tersamarkan visibility-nya.    * Pengguna dapat keluar (logout) dari halaman konten website dengan menekan tombol logout.   Spesifikasi Kebutuhan Sistem   1. Kebutuhan Fungsional    * Sistem menyediakan form login yang berisi email dan password yang harus diisi pengguna. Kemudian di sana juga terdapat tombol login. Ketika tombol tersebut ditekan, pengguna yang sudah melengkapi email dan password dengan benar dapat mengakses konten halaman website.    * Pengguna dapat keluar dari konten halaman website dengan menekan tombol logout. 2. Kebutuhan Non Fungsional    * Ketika pengguna memasukkan email di dalam form email, sistem dapat memeriksa apakah inputan tersebut sudah sesuai dengan struktur email yang benar. Jika tidak sesuai, pengguna diberi pengingat bahwa belum mengisi form email dengan benar.    * Ketika pengguna memasukkan password di dalam form password, sistem dapat menyembunyikan setiap huruf yang diketikkan. Sehingga, pengguna akan merasa aman ketika mengetik passwordnya di tempat umum sekalipun.    * Ketika proses autentikasi tidak berhasil, sistem menampilkan informasi bahwa terjadi kesalahan pengisian data email atau password. 3. Antarmuka Pengguna (User interface)    * Perangkat lunak front-end: HTML dan CSS    * Perangkat lunak back-end: JavaScript 4. Antarmuka perangkat keras (Hardware interface)    * Komputer atau ponsel cerdas dengan browser yang sudah terinstal di dalamnya.   Rencana Pengembangan Sistem:  Dalam pengembangan sistem ke depannya, pengembang software akan menambahkan fitur sebagai berikut:   * Enkripsi password pada aplikasi autentikasi halaman website. * Pengecekan panjang password yang harus memiliki minimal 6 karakter. |

### Komunikasi Aplikasi dengan Komputer

Ketika seorang pengembang software menulis dan menjalankan sebuah kode, maka terjadilah proses konversi. Proses tersebut dibedakan menjadi dua yaitu **Compile**dan **Interpret**.

Compiler merupakan sebuah aplikasi yang bertugas untuk mengonversi source code yang kita buat menjadi bahasa mesin.

* **Compiler** menerjemahkan seluruh kode sumber menjadi kode mesin sebelum program dijalankan, menghasilkan berkas executable.
* **Interpreter** menerjemahkan dan menjalankan kode baris per baris tanpa menghasilkan berkas executable, sehingga eksekusi terjadi secara langsung.

### Problem Solving Menggunakan Mindset komputasional

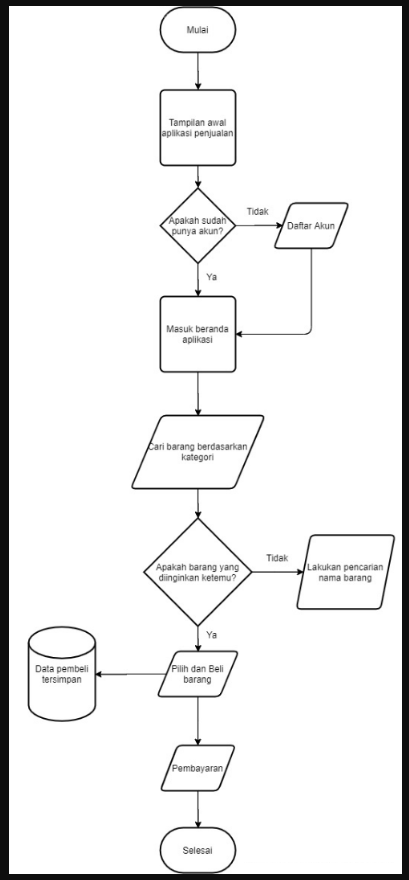
* 1. Pecah pecah masalahnya menjadi masalah kecil **( Decomposition )**
  2. membuat sesuatu “problem solving” sesuatu dapat melihat hasil contoh supaya paham konsep dan point2 untuk membangun sebuah aplikasi tersebut. **( Pattern Recognition )**
  3. Fokus ke hal penting atau umum saja. Jangan terlalu detail **( Abstraksi )**
  4. Menyusun langkah2 pengerjaan atau planning yang logis **( Algoritma )**
  5. Recheck **( Evaluasi )**.

### Flowchart

Yang sering dijumpai :



\*Skip forward\* Install aplikasi membuat flowchart & Buat Flowchart sederhana.



### Sikap Yang Baik Dengan TIM

Yang sering dijumpai :

* + 1. Komunikasi yang baik
    2. Membangun Trust
    3. Berikan semangat ( commend )
    4. Diskusi
    5. JANGAN Low Respond
    6. To do list, menentukan prioritas

### Rangkuman

* Spesifikasi Kebutuhan Aplikasi Perangkat Lunak adalah sebuah dokumen yang dibuat sebelum mengembangkan sebuah aplikasi perangkat lunak. Dokumen ini menjelaskan cara kerja dan kebutuhan fungsional maupun non-fungsional dari aplikasi yang digunakan pengguna nantinya.
* Beberapa keuntungan yang didapatkan dari pembuatan dokumen Spesifikasi Kebutuhan Aplikasi Perangkat Lunak:
  + Desainer dalam tim baik dari sisi UI/UX akan mendapat gambaran sehingga mereka dapat mencocokkan desain dengan kebutuhan aplikasi.
  + Tim penguji aplikasi (*tester*) akan mendapatkan panduan untuk membuat studi kasus untuk melakukan proses pengujian aplikasi.
  + Pengguna (*end user*) akan mendapatkan gambaran umum terkait aplikasi yang akan dibuat.
  + Pihak investor juga akan mendapatkan gambaran umum terkait fitur apa saja yang ada di dalam aplikasi. Sehingga membantu mereka untuk mengambil keputusan untuk investasi atau tidak.
* Cakupan elemen yang ada dalam dokumen Spesifikasi Kebutuhan Perangkat Lunak:
  + Tujuan dari aplikasi
  + Deskripsi umum mengenai aplikasi
  + Kebutuhan fungsional dan non fungsional dari aplikasi
  + Performa aplikasi dalam proses produksi
  + Antarmuka eksternal atau bagaimana sebuah aplikasi berinteraksi dengan perangkat keras atau perangkat lunak lainnya.
  + Batasan dari sistem aplikasi yang akan dibuat.
* Kebutuhan fungsional menggambarkan bagaimana sistem membantu menyelesaikan tugas atau kebutuhan pengguna ketika menggunakan aplikasi yang kita buat. Kebutuhan fungsional ini sama halnya dengan fitur yang ditawarkan aplikasi untuk pengguna. Tanpa terpenuhinya kebutuhan fungsional dalam dokumen SKPL, sama halnya dengan Anda meniadakan fungsi atau fitur dari sebuah aplikasi.
* Kebutuhan non fungsional mendukung kebutuhan fungsional yang sudah ada. Tanpa adanya persyaratan non fungsional, sistem aplikasi masih akan tetap bisa berjalan untuk memenuhi kebutuhan yang diinginkan pengguna. Walaupun begitu, kebutuhan non fungsional juga tetap harus diperhatikan karena dapat mempengaruhi pengalaman pengguna (*user experience*) ketika menggunakan aplikasi. Biasanya persyaratan non fungsional ini, meliputi kinerja, keandalan, keamanan, dan kegunaan dari aplikasi itu sendiri.

**Struktur Penulisan Dokumen Spesifikasi Kebutuhan Perangkat Lunak (SKPL)**

Anda dapat mengacu pada standar IEEE 1998 untuk menulis dokumen Spesifikasi Kebutuhan Perangkat Lunak. Standar ini dapat Anda sesuaikan kembali bergantung pada kebutuhan aplikasi atau perangkat lunak yang ingin dikembangkan. Untuk lebih jelasnya, simak tabel berikut ini:

| Bab | Deskripsi |
| --- | --- |
| Pengantar | Menjelaskan secara singkat fungsi dari sistem yang dibuat dan cara kerja aplikasi. Selain itu juga menjelaskan tujuan dari aplikasi yang akan dibuat. |
| Glosarium | Berisi definisi istilah-istilah teknis yang ada dalam dokumen. |
| Kebutuhan Pengguna | Menjelaskan secara umum layanan yang disediakan sistem untuk pengguna. Deskripsi yang ada bisa menggunakan narasi biasa atau diagram supaya dapat dimengerti. |
| Sistem Arsitektur | Menjelaskan gambaran tingkat tinggi (*high level overview*) dari sistem arsitektur yang digunakan dalam aplikasi. |
| Spesifikasi Kebutuhan Sistem | Menjelaskan secara lebih rinci tentang kebutuhan fungsional dan non fungsional dari aplikasi yang akan dibuat. Selain itu jika sistem memiliki antarmuka dengan sistem atau perangkat keras lain juga dapat ditambahkan di sini. |
| Model Sistem | Menjelaskan object model, data-flow models, atau semantic data models yang digunakan dalam aplikasi. |
| Rencana Pengembangan Sistem | Menjelaskan asumsi Anda terkait rencana pengembangan sistem di masa depan. Bagian ini akan berguna bagi desainer aplikasi untuk tidak membatasi idenya untuk kemungkinan pengembangan sistem. |
| Apendiks | Menjelaskan detail tambahan yang mendukung pengembangan aplikasi |

**Alur Kerja Aplikasi**

* Bahasa pemrograman tidak bisa langsung berkomunikasi dengan komputer. Komputer hanya bisa memahami bahasa mesin. Oleh karena itu diperlukan jembatan yaitu compiler atau interpreter supaya bahasa pemrograman bisa dipahami oleh komputer. Selengkapnya mengenai perbedaan compiler bisa Anda simak di tabel berikut:

| **Kategori** | **Compiler** | **Interpreter** |
| --- | --- | --- |
| Penggunaan | * + Source code telah dikonversi menjadi machine code.   + Waktu eksekusi program lebih singkat. | * + Lebih mudah digunakan untuk pemula yang baru belajar. |
| Hasil keluaran | Menghasilkan aplikasi atau program luaran berupa berkas executable (.exe) | Tidak menghasilkan aplikasi atau program luaran berupa berkas executable. |
| Efektifitas | Hasil kompilasi dari source code akan berjalan lebih cepat. | Berjalan lebih lambat ketika dieksekusi. |
| Platform | Spesifik ke platform tertentu (misal .exe tidak dapat dibuka di Mac) | Cross platform (asalkan memiliki interpreter yang sesuai) |
| Alur pembacaan | * + Compiler menampilkan pesan eror ketika terjadi kesalahan kode.   + Program tidak berjalan atau tidak menghasilkan berkas executable apabila kesalahan kodenya belum diperbaiki.   + Misal ada kode yang salah di baris ke-5, maka harus diperbaiki dulu. | * + Membaca satu per satu baris kode yang ada.   + Jika terdapat kode yang salah, interpreter menampilkan pesan eror dan harus diperbaiki untuk melanjutkan eksekusi baris selanjutnya.   + Misalnya terdapat eror di baris ke-5, program masih bisa berjalan karena baris pertama sampai keempat tidak ada eror. |
| Bahasa Pemrograman | C, C++, C#, Swift, Java | JavaScript, Python, PHP, Ruby. |

**Penyelesaian Masalah Menggunakan Cara Berpikir Komputasional**

Berikut teknik penyelesaian masalah menggunakan cara berpikir komputasional:

* **Memecah permasalahan yang besar menjadi bagian kecil (*decomposition*)**Dengan teknik decomposition tersebut, maka kita tidak lagi melihat sebuah permasalahan sebagai hal yang kompleks. Karena kita telah memecahnya menjadi bagian atau tugas-tugas kecil yang bisa dikerjakan.
* **Pelajari pola dari setiap permasalahan (*pattern recognition*)**Pattern Recognition merupakan teknik pemecahan masalah dengan melihat perbedaan atau persamaan pola dari berbagai permasalahan. Sehingga kita dapat memprediksi atau memproyeksikan solusi apa yang harus kita lakukan. Ketika kita dihadapkan dengan banyak permasalahan, kita harus bisa mempelajari polanya supaya dapat menyelesaikan dengan cepat.
* **Mengabstraksikan suatu permasalahan (*abstraction*)**Abstraksi merupakan cara berpikir komputasional yang identik dengan penyaringan dan pengumpulan data yang bersifat umum. Selain itu, abstraksi juga akan mengesampingkan detail data yang bersifat khusus atau yang tidak kita perlukan untuk lebih berkonsentrasi terhadap apa yang akan dilakukan.
* **Susun langkah menggunakan algoritma (*algorithm*)**Algoritma merupakan sekumpulan alur instruksi yang berurutan untuk menyelesaikan permasalahan. Sehingga kita akan merangkai sekumpulan instruksi tersebut menjadi urutan yang terstruktur, logis, dan mudah dipahami. Algoritma yang dibuat harus memuat beberapa hal yaitu, titik awal, titik akhir, dan serangkaian instruksi yang jelas.
* **Mengevaluasi solusi yang didapatkan (*evaluation*)**Proses terakhir yang dapat kita lakukan adalah melakukan evaluasi terhadap solusi yang telah kita dapatkan. Proses ini perlu dilakukan sebelum melangkah ke permasalahan atau tugas yang baru.

**Flowchart**

* Flowchart atau bisa disebut dengan diagram alur merupakan bentuk penggambaran dengan pendekatan visual terkait langkah-langkah dan keputusan untuk melakukan sebuah proses, alur kerja, ataupun algoritma. Beberapa jenis flowchart antara lain, process, workflow, dan program flowchart.
* Flowchart atau diagram alur memiliki beberapa simbol tertentu untuk mewakili fungsi atau langkahnya dalam sebuah proses. Garis dan panah menunjukkan urutan langkah dan hubungan antar blok diagram. Berikut adalah beberapa simbol yang sering dijumpai pada flowchart:

| Simbol | Nama | Fungsi |
| --- | --- | --- |
|  | Terminator | Titik awal atau akhir dari sebuah alur atau proses. |
|  | Proses | Menunjukkan proses, aktivitas, atau langkah. |
|  | Data | Menunjukkan input dan output. |
|  | Decision | Menunjukkan adanya pemilihan keputusan/decision making. |
|  | Arrow | Menghubungkan dari satu bentuk simbol ke simbol lainnya. Menunjukkan arah proses atau alur. |
|  | Database | Merepresentasikan dari penyimpanan data atau database. |
|  | Document | Merepresentasikan bentuk dokumen atau laporan. |

**Tips Sikap Kerja dalam Kolaborasi Perencanaan Aplikasi dengan Tim**

Untuk mengetahui tips sikap kerja dalam kolaborasi perencanaan aplikasi dengan tim, simak uraiannya di bawah ini:

* **Efektifkan Komunikasi**Dalam proses kolaborasi pasti tidak pernah lepas dari diskusi timbal balik alias komunikasi. Komunikasi yang efektif antar anggota tim akan membuat informasi tersampaikan dengan baik. Tanpa adanya komunikasi maka segala informasi akan tak tersampaikan dan kolaborasi pun tidak akan berjalan dengan semestinya.
* **Bertanggung jawab pada tugas yang dikerjakan**Membangun kepercayaan dalam tim juga berperan penting dalam keberhasilan kolaborasi. Mengapa demikian? Kepercayaan berarti Anda mengandalkan orang lain untuk melakukan sebuah pekerjaan. Anda percaya pada integritas dan kemampuan orang tersebut walaupun ada kemungkinan resiko yang bisa kembali ke diri Anda sendiri. Begitu pula sebaliknya, jika orang lain mengandalkan Anda untuk membantu pekerjaannya, maka orang tersebut percaya pada kemampuan dan integritas Anda.
* **Berikan Semangat Anda**Kita telah mengetahui pentingnya membangun kepercayaan dalam sebuah proses kolaborasi. Selain itu kita juga harus selalu menunjukkan semangat dan bergairah dalam mengerjakan tugas. Semangat dapat mempengaruhi mental dari tim. Maka dari itu ketika ada anggota tim yang sedang terlihat lesu karena kehilangan semangat atau bahkan hampir putus asa, kita harus menyemangatinya. Mungkin terdengar sepele tetapi hal tersebut akan mempengaruhi psikis anggota tim.
* **Utamakan Diskusi**Berpikir bersama dalam sebuah diskusi merupakan hal yang sangat penting dalam sebuah tim. Ketika muncul kendala yang cukup besar, diskusi bersama merupakan hal yang tepat untuk diterapkan. Jika rekan tim Anda berusaha menyelesaikan sendiri-sendiri, ajaklah mereka berdiskusi untuk mencapai tujuan bersama. Utarakan pendapat Anda sesuai dengan tujuan yang ingin dicapai. Anda juga harus bisa mendengarkan dan menghargai pendapat orang lain ketika mereka mengemukakan idenya.
* **Hindari Sikap Tidak Tanggap (*Slow Response*)**Kondisi tidak tanggap atau *slow response* ini biasanya sering ditemui ketika bekerja secara jarak jauh. Satu-satunya cara untuk saling berkomunikasi adalah dengan mengirim pesan atau melakukan *video call*. Sehingga anggota tim yang bekerja secara jarak jauh harus selalu *stand by* apabila sewaktu-waktu ada hal penting yang harus segera dikerjakan.
* **Menentukan Prioritas**Kita pasti akan menemui kondisi di mana banyak kerjaan atau todo yang harus kita kerjakan dalam satu waktu. Dan terkadang semua todo adalah hal penting yang memang harus dikerjakan. Nah, dari permasalahan tersebut kita perlu menentukan prioritas terhadap pekerjaan kita.  
    
  Berikut rekomendasi yang bisa Anda ikuti dalam menentukan prioritas pekerjaan:
  + Kumpulan semua todo dalam satu tempat
  + Identifikasi todo yang urgent (harus segera diselesaikan) vs important (penting untuk dikerjakan)
  + Perhitungkan nilai tiap todo
  + Urutkan todo berdasarkan usaha yang harus dikeluarkan
  + Fleksibel dan Mudah Beradaptasi
  + Hapus todo yang bisa dihapus

# BAB 4 – Konsep Dasar Pemograman

Berikut adalah pokok pembahasan yang akan diulas dalam materi konsep dasar pemrograman:

* Pertama kita akan belajar sintaksis. Di sana kita akan belajar case sensitive, menulis statement, whitespace, dan keyword. Selain itu Anda juga akan mengenal kode semu (pseudocode) dan cara membuatnya.
* Setelah itu Anda akan mempelajari elemen-elemen dasar yang ada di dalam kode pemrograman, seperti variabel, operator, dan tipe data.
* Kemudian Anda juga akan belajar logika pemrograman, seperti logika perbandingan dan perulangan.
* Terakhir, Anda akan dikenalkan dengan bahasa pemrograman JavaScript serta penerapan konsep dasar pemrograman dalam bahasa tersebut.

## Glosarium ?

-----------------

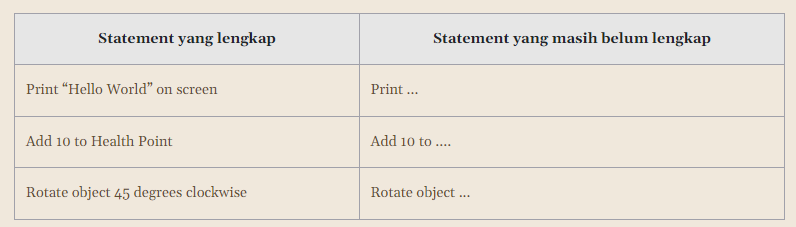
### Syntax

**Syntax** adalah aturan atau tata cara penulisan kode dalam bahasa pemrograman tertentu.

Tiap bahasa pemograman punya beda2 cara penulisan.

* Case Sensitive, pada syntax adalah beda sedikit pada pemanggilan kode maka akan error. Contohnya huruf kapital “true & True”.
* Non sensitive, kebalikannya.

### Statement

Instruksi Tunggal. “ Rev kendaraan tersebut ke limitnya “  


### Whitespace

Spasi, tab, atau enter. Yang mengakibatkan kosongnya suatu “baris ?” ya contohnya “ “ kosong.

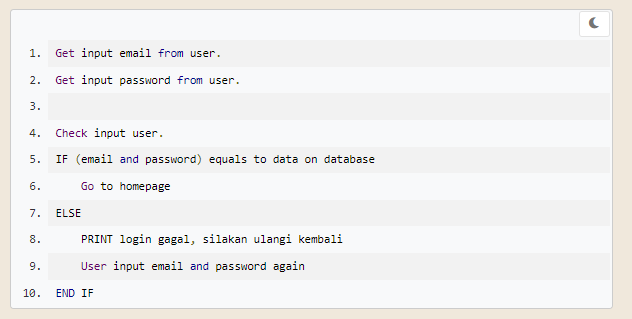
**Mostly** Kebanyakan bahasa pemograman gak terlalu sensitive sama *Whitespace* ini, namun sangat membantu untum melihat kerapihan kode bagi para developer.

### Keyword

Keyword, adalah kata2. **Bedanya dengan SYNTAX** adalah. Syntax itu cara penulisan.

### Pseudocode

Pseucode tidak ada standar khusus dalam hal penulisan. Namun, kita harus tetap membuatnya sederhana dan mudah dipahami.



Jadi, di sini tidak ada hal khusus yang harus diperhatikan, yang terpenting adalah alur dan isi dari yang kita buat. Oke?

Jadi. Kalian pretend aja kaya nulis kode ( adanya if, else, dll. Bahasa inggris lah ), Tapi harus masih bisa dibaca sama orang biasa ( tidak terlalu teknis )

### Variable

Terdapat tiga hal yang harus diperhatikan dalam sebuah variabel yaitu nama, nilai, dan tipe dari variabel.

* Variable adalah wadah untuk menyimpa**n suatu data terkait konteks yang diciptakan oleh penciptanya**. Contoh :
  + Nama variable : Makanan
    - Tipe nya : String
      * JADI : String Makanan = “Mie ayam, Nasi Padang, Pecel”
    - Cara bacanya, variable makanan yang tipenya string akan berisi kata2 atau nama makanan yang enak.

Nanti, **JANGAN MENULIS NAMA VARIABEL YANG SAMA DENGAN KEYWORD**

### Tipe Data

**CARI SENDIRI LAH DI INTERNET TIPE DATA TUH APA AJA.**

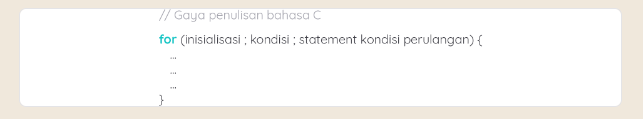
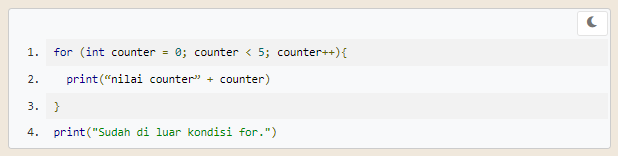
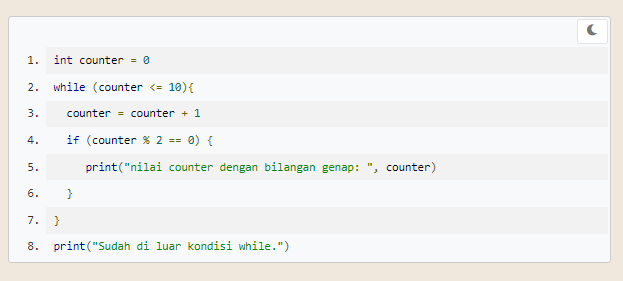
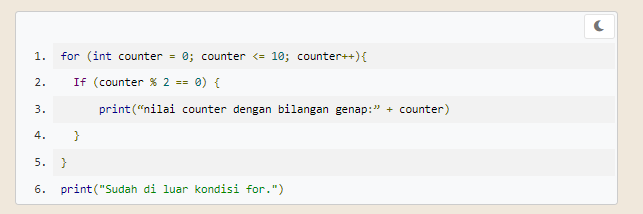
### Perbandingan ( IF\_Else )

1. Ada yang =, !=, ==,
2. Ada yang, &&
3. Ada yang >, <, >=

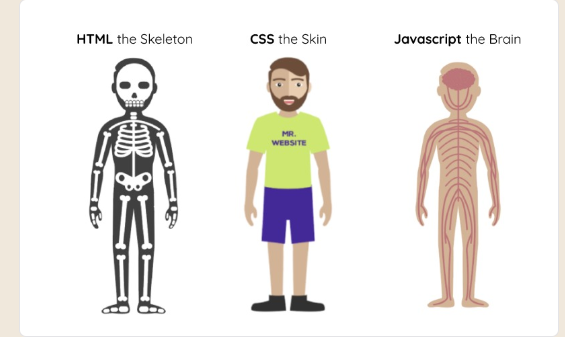
You know lah, cari di internet.

Setertusnya itu hanya contoh if else, dari **bahasa C**. Ini bisa berbeda2 di setiap bahasa.

### LOOP

* While :
  + While, ya ketika. Simple
    - Mobil = 0
    - ( Ketika sebuah mobil < 5, maka mobil +1 )
    - Mobil akan tergenerate hingga dibawah 5.
* **For** :
  + bedanya dengan while? While digunakan ketika **kita tidak mengetahui berapa kali perulangan akan berjalan.**
    - **1. **
    - **2. **
    - Bagian pertama adalah inisialisasi variabel counter yang bernilai 0.
    - Kemudian terdapat kondisi counter < 5 yang menandakan perulangan variabel counter tidak sampai 5.
    - Setelah itu bagian ketiga adalah akhiran yang berupa statement. Masih ingat dengan statement pada while yang berisi counter = counter+1?
* **Gabungkan IF dengan LOOP.**
  + ****
  + ****
    - Susunannya.
      * 1. Inisialisasi Variable
      * 2. Loop
      * 3. Didalam loop berikan jika maka ( IF / Else ).
      * 4. Print atau tampilan hanya di if else

# BAB 4.2 – Javascript ( Website )



Yup, Website kita sekarang BOS.