# BAB II
# TINJAUAN PUSTAKA

## 2.1 Profil Perusahaan

Berikut adalah profil dari perusahaan tempat kami melakukan kerja praktik.

### 2.1.1 Gambaran Umum Perusahaan

CV. Metric Design adalah sebuah perusahaan penyedia jasa pembuatan perangkat lunak. Berlokasi di Jalan Rereng Adumanis No 51 Bandung, CV. Metric Design didirikan pada tahun 2006 dimana pendirinya merupakan alumni yang berasal dari perguruan tinggi yang sama yaitu ITB. Para pendiri CV. Metric Design terdiri dari berbagai jurusan yang berbeda namun memiliki keinginan yang sama pada bisnis, terutama dibidang teknologi.

Pada awalnya CV. Metric Design hanya menangani pekerjaan yang terbatas pada desain saja. Namun seiring dengan meningkatnya kebutuhan klien dan pasar maka CV. Metric Design mulai memasuki bidang multimedia, dimana CD interaktif merupakan *core business* dari CV. Metric Design pada saat itu. Seiring banyaknya pekerjaan dari lembaga pemerintahan dan perusahaan besar yang membutuhkan data dan kelengkapan administrasi yang lebih tersruktur maka CV. Metric Design membuat badan hukum pada awal 2008.

Pada tahun 2009 CV. Metric Design mulai merambah pada bisnis baru seperti pembuatan perangkat lunak untuk perangkat *mobile* baik untuk kebutuhan internal perusahaan maupun untuk kebutuhan eksternal seperti pelanggan perusahaan tersebut. Sampai saat ini CV. Metric Design telah mengerjakan lebih dari 50 perangkat lunak dengan skala pekerjaan dan klien yang cukup besar, sebut saja Pertamina, Telkom, Lintas Artha, Sampoerna, Suzuki, dan lainnya. Tidak hanya itu CV. Metric Design juga telah memulai untuk mengembangkan produk sendiri diberbagai media, seperti website, perangkat *mobile*, dan media lainnya.

### 2.1.2 Visi & Misi Perusahaan

Berikut adalah visi misi dari perusahaan tempat kami melakukan kerja praktik.

- **VISI**

  To be an Indonesian company leader at digital industry.

- **MISI**

  1) Build a strong and coherence teamwork.
  2) Generate multilevel partnership network market.
  3) Best management quality system.
  4) Smooth and efficient management execution.
  5) Make strong financial management.

### 2.1.3 Sejarah Perusahaan

Metric adalah sebuah perusahaan yang bergerak dibidang multimedia, yang berlokasi di jalan rereng adumanis no 51 Bandung. Metric didirikan pada tahun 2006, dimana pendirinya adalah alumni dari sabuah perguruan tinggi yang sama yaitu ITB yang terdiri dari berbagai jurusan yang berbeda namun memiliki keinginan yang sama pada bisnis, terutama dibidang teknologi.

Pada awal mulanya metric hanya menangani pekerjaan yang bersifat desain saja. Namun lama kelamaan seiring dengan meningkatnya kebutuhan klien/pasar maka mtric mulai memasuki bidang multimedia, dimana CD interaktif merupakan core dari bisnis metric. Siiring banyaknya pekerjaan dari lembaga pemerintahan dan perusahaan besar, membutuhkan data dan kelengkapan administrasi yang lebih tersruktur, maka metric membuat Badan hukum pada awal 2008

Pada tahun 2009 metric mulai merambah ke newmedia seperti mobile device dan touchscreen baik untuk kebutuhan internal perusahaan/instansi maupun untuk kebutuhan pelanggan perusahaan tersebut. Sampai saat ini metric telah mengerjakan lebih dari 50 aplikasi multimedia dengan skala

pekerjaan dank lien yang cukup besar, sebut saja Pertamina, Telkom, Lintas artha, Sampoerna, Suzuki, dan lainnya

Saat ini metric tidak hanya mengerjakan perkerjaan dari klien, akan tetapi mulai mengembangkan produk sendiri diberbagai media, seperti web, mobile device, dan newmedia lainnya.

### 2.1.4 Logo Perusahaan

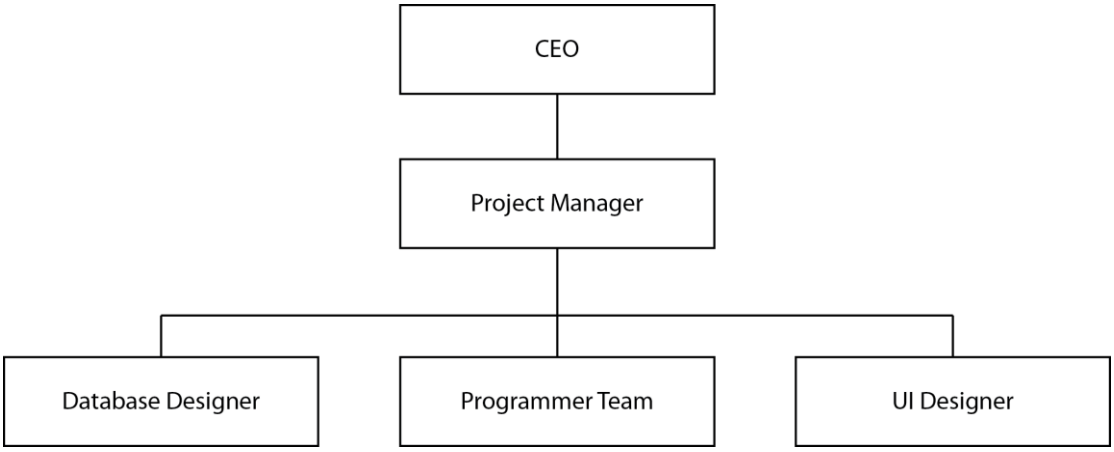
Berikut adalah Logo Dari CV. Metric Design




Gambar 2. Logo CV. Metric Design

### 2.1.5 Struktur Organisasi Perusahaan

Pada dasarnya struktur organisasi dari CV. Metric Design belum terbentuk seperti perusahaan pada umumnya. Namun terdapat struktur organisasi yang langsung menangani proyek seperti terlihat pada Gambar 3.




Gambar**Error! No text of specified style in document.**3 Struktur Organisasi CV. Metric Design

### 2.1.6 Deskripsi Jabatan

Berikut keterangan dari struktur organisasi tim pengembang Metric-Design pada pada Gambar 1:

1. CEO, merupakan kepala dari CV. Metric Design.
2. *Project Manager*, bertanggung jawab dalam pembuatan perangkat lunak beserta pembagian tugas dan waktu kerja.
3. *Database Designer*, bertugas merancang basis data dan bertanggung jawab atas segala hal yang berkaitan dengan DBMS.
4. *Programmer*, bertugas pada pembangunan aplikasi meliputi pembuatan perangkat lunak sesuai dengan rancangan sampai dengan pengujian perangkat lunak itu sendiri.
5. UI *Designer*, bertugas membuat rancangan antarmuka dari perangkat lunak yang akan dibangun.

## 2.2 Landasan Teori

Berikut adalah landasan teori dari pembangunan aplikasi kami.

### 2.2.1 Kebutuhan Non Fungsional

Batasan-batasan dari layanan-layanan dan fungsi-fungsi dari sebuah sistem, seperti: batasan waktu, batasan dari pengembangan proses, dan batasan pengguna. Analisis kebutuhan non fungsional pada sistem ini

meliputi analisis perangkat lunak, analisis perangkat keras, analisis jaringan, perangkat pikir dan analisis pengkodean.[23]

### 2.2.2 Flowmap

Flowmap adalah campuran peta dan flow chart, yang menunjukkan pergerakan benda dari satu lokasi ke lokasi lain, seperti jumlah orang dalam migrasi, jumlah barang yang diperdagangkan, atau jumlah paket dalam jaringan. Flowmapmenolong analis dan programmer untuk memecahkan masalah kedalam segmen-segmen yang lebih kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam pengoperasian.[24]

### 2.2.3 Data

Data merupakan bahan baku informasi, dapat didefinisikan sebagai kelompok teratur simbol-simbol yang mewakili kuantitas, fakta, tindakan, benda dan sebagainya. Data terbentuk dari karakter, dapat berupa alphabet, angka, maupun simbol khusus seperti *,$ dan /. Data disusun mulai dari *bits, bytes, fields, records, file* dan *database* [1].

#### 2.2.3.1 Entitas

Entitas adalah objek di dunia yang bersifat unik walaupun tidak harus dalam bentuk fisik. Setiap entitas mempunyai atribut yang membedakannya dengan entitas lainnya. Contoh: entitas Mahasiswa, Dosen, Mobil, dan lain sebagainya[2].

#### 2.2.3.2 Asumsi

Asumsi adalah kepercayaan, gagasan, dugaan, atau pemikiran yang dimiliki oleh Anda, sekelompok orang, atau para ahli internal atau eksternal mengenai suatu objek. Asumsi-asumsi ini menjadi penting karena kita menggunakan asumsi kita ini sebagai panduan bagi tindakan dan pengambilan keputusan yang kita lakuan[3].

#### 2.2.3.3 ERD (Entity Relational Diagram)

ERD (Entity Relationship Diagram) adalah suatu model jaringan yang menggunakan susunan data yang disimpan dalam sistem secara abstrak. ERD merupakan model jaringan data yang menekankan pada struktur dan hubungan antardata[4].

Terdapat beberapa notasi simbol pada ERD yaitu:

1. Persegi Panjang, menyatakan himputan entitas.
2. Oval, menyatakan atribut (atribut key digaris bawahi).
3. Belah ketupat, menyatakan himpunan relasi.
4. Garis, menyatakan penghubung antara himpunan relasi dengan himpunan entitas dan himpunan entitas dengan atributnya.

#### 2.2.3.4 Kardinalitas

Kardinalitas atau derajat relasi menunjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain. Macam-macam kardinalitas adalah:

1. **satu ke satu (one to one),** Setiap anggota entitas A hanya boleh berhubungan dengan satu anggota entitas B, begitu pula sebaliknya.
2. **satu ke banyak (one to many),** setiap anggota entitas A dapat berhubungan dengan lebih dari satu anggota entitas B tetapi tidak sebaliknya.
3. **Banyak ke banyak (many to many),** setiap entitas A dapat berhubungan dengan banyak entitas himpunan entitas B dan demikian pula sebaliknya[2].

#### 2.2.3.5 Skema Relasi

Skema relasi merupakan deskripsi dari data dalam terminology model data. Pada model relational, skema untuk relasi akan menyebutkan nama dari setiap kolom (field) yang ada dan domain yang menyertainya.

### 2.2.4 *Structure Analysis and Design*

Struktur analisis dan design adalah sebuah metodologi yang di gunakan pada rekayasa perangkat lunak untuk mendeskripsikan sistem menjadi sebuah fungsi hirarki[5].

#### 2.2.4.1 Diagram Konteks

Diagram Konteks adalah diagram yang menggambarkan sebuah sistem secara menyeluruh. yang selanjutnya akan lebih di detail kan oleh DFD. Konteks diagram juga biasa disebut sebagai DFD level 0. Contoh gambar diagram konteks.

#### 2.2.4.2 Data Flow Diagram

Data Flow Diagram atau biasa di singkat DFD merupakan serangkai diagram yang menggambarkan kegiatan-kegiatan yang ada dalam suatu sistem. Teknik pembuatan DFD dimulai dengan menggambarkan sistem secara global dan dilanjutkan dengan analisis masing-masing bagian. Pada awalnya digambarkan konteks diagram yang menggambarkan sebuah sistem secara menyeluruh yang akan diinvestigasi. Konteks diagram tersebut dapat dikatakan sebagai DFD level 0. Analisis sistem yang lebih detail selanjutnya dapat dilakukan dengan menggambarkan DFD level 1,2 dan seterusnya.

#### 2.2.4.3 Spesifikasi Proses

Spesifikasi Proses merupaka tabel yang berisi keterangan deskripsi dari semua proses yang terdapat di DFD. Logika proses harus dituliskan secara jelas baik menggunakan bahasa deskriptif atau pseudo code (tidak boleh campuran.

#### 2.2.4.4 Kamus Data

Kamus Data (Data Dictionary) merupakan fakta tentang data dan kebutuhan-kebutuhan informasi dari sistem informasi. Dengan menggunakan data dictionary, analis sistem dapat mendefinisikan data

yang mengalir dalam sistem dengan lengkap. Tahap analisis dapat digunakan sebagai alat komunikasi antara anlisis sistem dengan pemakai sistem tentang data yang mengalir di sistem, yaitu tentang data yang masuk ke sistem dan tentang informasi yang dibutuhkan oleh pemakai sistem[6]. Contoh Kamus Data:

### 2.2.5 *Object Oriented Analysis and Design*

Analisis dan desain berorientasi objek adalah cara baru dalam memikirkan suatu masalah dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata. Dasar pembuatannya adalah objek, yang merupakan kombinasi antara struktur data dan perilaku dalam satu entitas. Pengertian "berorientasi objek" berarti bahwa kita mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya.

Konsep OOAD mencakup analisis dan desain sebuah sistem dengan pendekatan objek, yaitu analisis berorientasi objek (OOA) dan desain berorientasi objek (OOD). OOA adalah metode analisis yang memberikan requirement (syarat/keperluan) yang harus dipenuhi sebuah sistem dari sudut pandang kelas-kelas dan objek-objek yang ditemui dalam ruang lingkup perusahaan. Sedangkan OOD adalah metode untuk mengarahkan arsitektur software yang didasarkan pada manipulasi objek-objek sistem atau subsistem.

Hasil analisis berorientasi objek adalah deskripsi dari apa sistem secara fungsional diperlukan untuk melakukan, dalam bentuk sebuah model konseptual. Itu biasanya akan disajikan sebagai seperangkat menggunakan kasus, satu atau lebih UML diagram kelas, dan sejumlah diagram interaksi. Tujuan dari analisis berorientasi objek adalah untuk mengembangkan model yang menggambarkan perangkat lunak komputer karena bekerja untuk memenuhi seperangkat persyaratan yang ditentukan pelanggan[7].

#### 2.2.5.1 UML (Unified Modeling Language)

*Unified Modelling Language* (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C. Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan syntax /semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML syntax mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (Object-Oriented Design), Jim Rumbaugh OMT (Object Modeling Technique), dan Ivar Jacobson OOSE (Object-Oriented Software Engineering)[8].

Dalam membangun block UML ada 3 hal yang harus diperhatikan, yakni *Object* (memodelkan konsep), *Relationship* (Mengkoneksikan *Object*), dan Diagram (grouping yang saling mengkoneksikan antara *Object* dan *Relationship.*

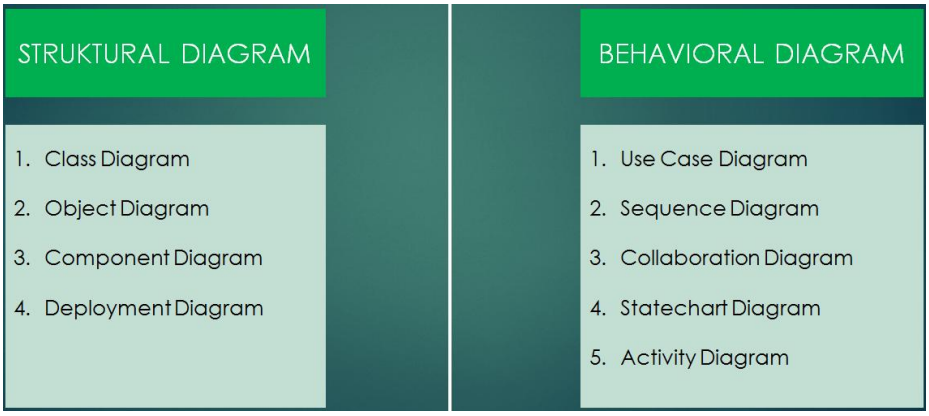
| UML Elements | Spesific UML Details |
|---|---|
| **Structural Things** | Classes |
| | Interfaces |
| | Collaborations |
| | Use Cases |
| | Active Classes |
| | Components |
| | Nodes |
| **Behavioral Things** | Interactions |
| | State Machines |
| **Grouping Things** | Packages |
| **Annotational Things** | Notes |

Gambar 4. Komponen *Object*

| UML Elements | Spesific UML Details |
|---|---|
| **Structural Relationships** | Dependencies |
| | Aggregations |
| | Associations |
| | Generalizations |
| **Behavioral Relationships** | Communicates |
| | Includes |
| | Extends |
| | Generalizes |

Gambar 5. Komponen *Relationship*

Tipe diagram UML ada 2 macam, yakni *Structural Diagram* (digunakan untuk mendeskripsikan relasi antar kelas) & *Behavioral Diagram* (digunakan untuk mendeskripsikan interaksi antara aktor dan sebuah *usecase* bagaimana aktor menggunakan sistem).



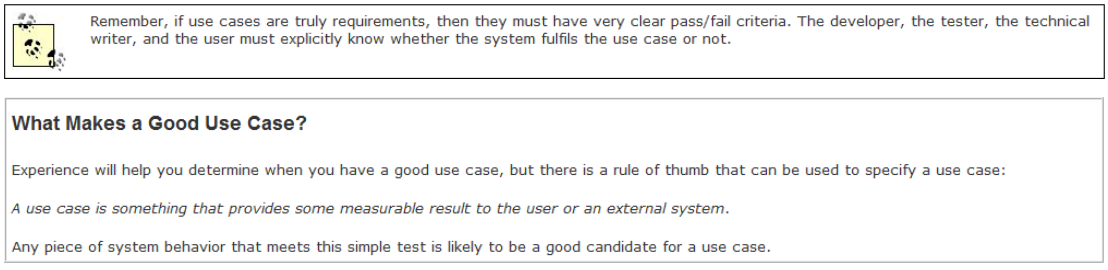

Gambar 6. Komponen Diagram

Diagram-diagram yang umum dipakai dalam analisis dan desain adalah diagram-diagram berikut,

### 2.2.5.2 Usecase Diagram

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah use case merepresentasikan sebuah interaksi antara aktor dengan sistem. Use case merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng- create sebuah daftar belanja, dan sebagainya. *Seorang/sebuah aktor* adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. Use case diagram dapat sangat membantu bila kita sedang menyusun requirement sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang test case untuk semua feature yang ada pada sistem. Sebuah use case dapat meng- include fungsionalitas use case lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa use case yang di- include akan dipanggil setiap kali use case yang meng- include dieksekusi secara normal. Sebuah use case dapat di- include oleh lebih dari satu use case lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang common . Sebuah use case juga dapat meng- extend use case lain dengan behaviour -nya sendiri. Sementara hubungan generalisasi antar use case menunjukkan bahwa use case yang satu merupakan spesialisasi dari yang lain.

Berikut adalah dasar menentukan suatu usecase,




Gambar 7. Dasar Pembuatan Usecase

Sebuah use case adalah sesuatu yang menyediakan beberapa hasil terukur kepada pengguna atau sistem eksternal. Use case harus memiliki sangat jelas kriteria lulus / gagal. Pengembang, tester, penulis teknis, dan pengguna harus secara eksplisit tahu apakah sistem memenuhi kasus

penggunaan atau tidak. Setiap bagian dari usecase yang memenuhi tes sederhana ini mungkin menjadi kandidat yang baik untuk usecase[9].

### 2.2.5.3 Usecase Scenario

Sebuah diagram yang menunjukkan Use case dan aktor mungkin menjadi titik awal yang bagus, tetapi tidak memberikan detail yang cukup untuk desainer sistem anda untuk benar-benar memahami persis bagaimana sistem dapat terpenuhi. Bagaimana bisa seorang desainer sistem memahami siapa aktor yang paling penting adalah dari notasi use case saja? Langkah-langkah apa yang terlibat dalam kasus penggunaan? Cara terbaik untuk mengungkapkan informasi penting ini adalah dalam bentuk penggunaan usecase scenario berbasis teks perusecasenya. Berikut adalah dasar format penulisan usecase scenario[9].

| **Use case description detail** | **What the detail means and why it is useful** |
|---|---|
| Related Requirements | Some indication as to which requirements this use case partially or completely fulfills. |
| Goal In Context | The use case's place within the system and why this use case is important. |
| Preconditions | What needs to happen before the use case can be executed. |
| Successful End Condition | What the system's condition should be if the use case executes successfully. |
| Failed End Condition | What the system's condition should be if the use case fails to execute successfully. |
| Primary Actors | The main actors that participate in the use case. Often includes the actors that trigger or directly receive information from a use case's execution. |
| Secondary Actors | Actors that participate but are not the main players in a use case's execution. |
| Trigger | The event triggered by an actor that causes the use case to execute. |
| Main Flow | The place to describe each of the important steps in a use case's normal execution. |
| Extensions | A description of any alternative steps from the ones described in the Main Flow. |

Gambar 8. Dasar Pembuatan Usecase Scenario

Contoh,

| **Use case name** | **Create a new Blog Account** | |
|---|---|---|
| Related Requirements | Requirement A.1. | |
| Goal In Context | A new or existing author requests a new blog account from the `Administrator`. | |
| Preconditions | The system is limited to recognized authors and so the author needs to have appropriate proof of identity. | |
| Successful End Condition | A new blog account is created for the author. | |
| Failed End Condition | The application for a new blog account is rejected. | |
| Primary Actors | Administrator. | |
| | Secondary Actors | Author Credentials Database. |
| | Trigger | The `Administrator` asks the CMS to create `a new blog account`. |
| Main Flow | Step | Action |
| | 1 | The `Administrator` asks the system to create a new blog account. |
| | 2 | The `Administrator` selects an account type. |
| | 3 | The `Administrator` enters the author's details. |
| | 4 | The author's details are verified using the Author Credentials Database. |
| | 5 | The new blog account is created. |
| | 6 | A summary of the new blog account's details are emailed to the author. |
| Extensions | Step | Branching Action |
| | 4.1 | The Author Credentials `Database` does not verify the author's details. |
| | 4.2 | The author's new blog account application is rejected. |

Gambar 9. Contoh Usecase Scenario

### 2.2.5.4 Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display , dan sebagainya) berupa message yang digambarkan terhadap waktu. Sequence diagram terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang men- trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan. Masing-masing objek, termasuk aktor, memiliki lifeline vertikal. Message digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, message akan dipetakan menjadi operasi/metoda dari class. Activation bar menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message[8].

Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan icon khusus untuk objek boundary, controller dan persistent entity.

### 2.2.5.5 Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek[9]. Class menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). Class diagram menggambarkan struktur dan deskripsi class, package dan objek beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lain-lain. Class memiliki tiga area pokok :

1. Nama (dan stereotype)
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

1. Private , tidak dapat dipanggil dari luar class yang bersangkutan
2. Protected , hanya dapat dipanggil oleh class yang bersangkutan dan anak-anak yang mewarisinya
3. Public , dapat dipanggil oleh siapa saja

Class dapat merupakan implementasi dari sebuah interface , yaitu class abstrak yang hanya memiliki metoda. Interface tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah class. Dengan demikian interface mendukung resolusi metoda pada saat run-time. Sesuai dengan perkembangan class model, class dapat dikelompokkan menjadi package. Kita juga dapat membuat diagram yang terdiri atas package.

Hubungan Antar Class

1. Asosiasi, yaitu hubungan statis antar class . Umumnya menggambarkan class yang memiliki atribut berupa class lain, atau class yang harus mengetahui eksistensi class lain. Panah navigability m enunjukkan arah query antar class .
2. Agregasi, yaitu hubungan yang menyatakan bagian ("terdiri atas..").
3. Pewarisan, yaitu hubungan hirarkis antar class . Class dapat diturunkan dari class lain dan mewarisi semua atribut dan metoda class asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari class yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.

Hubungan dinamis, yaitu rangkaian pesan ( message ) yang di- passing dari satu class kepada class lain. Hubungan dinamis dapat digambarkan dengan menggunakan sequence diagram yang akan dijelaskan kemudian.

### 2.2.5.6 Activity Diagram

Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir. Activity diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. Activity diagram merupakan state diagram khusus, di mana sebagian besar state adalah action dan sebagian besar transisi di-

trigger oleh selesainya state sebelumnya (internal processing). Oleh karena itu activity diagram tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. Sebuah aktivitas dapat direalisasikan oleh satu use case atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara use case menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Sama seperti state , standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. Decision digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel ( fork dan join ) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. Activity diagram dapat dibagi menjadi beberapa object swimlane untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu[8].

#### 2.2.5.7 Deployment Diagram

Deployment/physical diagram menggambarkan detail bagaimana komponen di- deploy dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal Sebuah node adalah server, workstation , atau piranti keras lain yang digunakan untuk men- deploy komponen dalam lingkungan sebenarnya. Hubungan antar node (misalnya TCP/IP) dan requirement dapat juga didefinisikan dalam diagram ini[8].

### 2.2.6 Implementasi

Berikut adalah landasan teori dari tahap-tahap yang ada pada bagian implementasi.

#### 2.2.6.1 Database

*Database* adalah kumpulan data yang saling berelasi[10]. Data sendiri merupakan fakta mengenai objek, orang, dan lain-lain. Data dinyatakan dengan nilai (angka, deretan karakter, atau symbol). Basis data dapat didefinisikan dalam berbagai sudut pandang seperti berikut :

1. Himpunan kelompok data yang saling berhubungan yang diorganisasi sedemikian rupa sehingga kelak dapat dimanfaatkan dengan cepat dan mudah.
2. Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa tanpa pengulangan (redudancy) yang tidak perlu, untuk memenuhi kebutuhan.

   Kumpulan file/table/arsip/ yang saling berhubungan yang disimpan dalam media penyimpan elektronik.

#### 2.2.6.2 Mysql

Mysql adalah sebuah perangkat lunak sistem menajemen basis data SQL atau yang dikenal dengan DBMS (*Database Management System)*, database ini *multithread, multi-user*. MySQL sangat handal yang awalnya SQL berfungsi sebagai bahasa penghubung antara program database dengan bahasa pemrograman[11].

#### 2.2.6.3 PHP

PHP (Hypertext Preprocessor) yaitu bahasa pemrograman web-server side yang bersifat open source. PHP merupakan script yang terintegrasi dengan HTML dan berada pada server (server side HTML embedded scripting). PHP adalah script yang digunakan untuk membuat halaman website yang dinamis. Dinamis berarti halaman yang akan ditampilkan dibuat saat halaman itu diminta oleh client. Mekanisme ini menyebabkan informasi yang diterima client selalu yang terbaru. Semua script PHP dieksekusi pada server dimana script tersebut dijalankan[12].

#### 2.2.6.4 HTML & CSS

*HTML* (*Hyper Text Markup Language*) merupakan metode yang menautkan (*link*) satu dokumen ke dokumen lain melalui suatu teks. HTML merupakan kumpulan kode-kode yang ditampilkan dalam format tag untuk ditampilkan dalam halaman Web[13].

*CSS (Cascading Style Sheet)* adalah sebuah dokumen yang berdiri sendiri dan dapat dimasukkan dalam kode HTML atau sekedar menjadi rujukan oleh HTML dalam pendefinisian style. CSS menggunakan kode-kode yang tersusun untuk menetapkan style pada elemen HTML atau dapat juga digunakan untuk membuat style baru yang biasa disebut class[14].

#### 2.2.6.5 Ajax & Jquery

*Ajax* adalah singkatan dari *Asynchronous Javascript and XML*. AJAX merupakan teknik baru didunia pengembangan web yang berguna untuk membuat aplikasi web, yang lebih "kaya" dibandingkan dengan aplikasi web biasa. AJAX memungkinkan sebuah halaman web memperbarui data dari server yang ditampilkan tanpa harus melakukan refresh, sehingga halaman terlihat lebih responsif[15].

*Jquery* adalah library atau kumpulan kode Javascript siap pakai. Keunggulan menggunakan jquery dibandingkan dengan javascript standar, yaitu menyederhanakan kode javascript dengan cara memanggil fungsi-fungsi yang disediakan oleh jquery. Javascript sendiri merupakan bahasa scripting yang bekerja disisi client/browser sehingga website bisa lebih interaktif[16].

#### *2.2.6.6 API*

API atau *Application Programming Interface* adalah sebuah protokol yang digunakan sebagai antar muka oleh komponen software untuk berkomunikasi dengan yang lainnya. Saat ini API banyak digunakan sebagai konten pembagi. Singkatnya, API dapat membolehkan para komunitas web untuk membuat sebuah API yang berguna untuk membagi

konten dan data diantara para komunitas lainnya. Dalam hal ini, konten yang dibuat dalam suatu tempat secara dinamik dapat dipost dan diupdate pada lokasi yang berbeda-beda pada web[17].

Contohnya seperti:

- Foto dapat dibagi dari website seperti flickr, photobucket ke sosial website seperti facebook dan myspace.
- Konten dapat di embed, seperti embed dari slideShare pada profil LinkedIn.
- Konten video dapat di embed pada website yang disediakan berlainan host.

API yang dimaksudkan untuk digunakan oleh lebih dari satu bahasa tingkat tinggi pemrograman sering memberikan, atau ditambah dengan fasilitas yang secara otomatis memetakan API untuk fitur (sintaksis atau semantik) yang lebih alami dalam bahasanya. Hal ini dikenal sebagai language binding, dan API itu sendiri. Tujuannya adalah untuk merangkum sebagian besar fungsi yang diperlukan dari API, meninggalkan lapisan tipis yang tepat untuk setiap bahasa.

### *2.2.6.7 JSON*

JSON (JavaScript Object Notation) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (generate) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemprograman JavaScript, Standar ECMA-262 Edisi ke-3 - Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data[18].

- JSON terbuat dari dua struktur:

  Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (object), rekaman (record), struktur (struct),

kamus (dictionary), tabel hash (hash table), daftar berkunci (keyed list), atau associative array.

Daftar nilai terurutkan (an ordered list of values). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (array), vektor (vector), daftar (list), atau urutan (sequence). Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemprograman moderen mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga berdasarkan pada struktur data ini.

- JSON menggunakan bentuk sebagai berikut:

1. Objek adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma).
2. Larik (*Array*) adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [ (kurung kotak buka) dan diakhiri dengan ] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma).
3. Nilai (value) dapat berupa sebuah string dalam tanda kutip ganda, atau angka, atau true atau false atau null, atau sebuah objek atau sebuah larik. Struktur-struktur tersebut dapat disusun bertingkat.
4. String adalah kumpulan dari nol atau lebih karakter Unicode, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan backslash escapes "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string. String sangat mirip dengan string C atau Java.
5. Angka adalah sangat mirip dengan angka di C atau Java, kecuali format oktal dan heksadesimal tidak digunakan.

Spasi kosong (whitespace) dapat disisipkan di antara pasangan tanda-tanda tersebut, kecuali beberapa detil encoding yang secara lengkap dipaparkan oleh bahasa pemprograman yang bersangkutan.

#### 2.2.6.8 Android

Android adalah sistem operasi untuk perangkat mobile yang pengembangannya dipimpin oleh Google. Sistem operasi ini bersifat *Open Source* dan dikembangkan berdasarkan kernel Linux yang disematkan pada gadget, baik itu handphone atau tablet[19].

### 2.2.7 *Flowchart*

Flowchart atau bagan alur merupakan metode untuk menggambarkan tahap-tahap penyelesaian masalah (prosedur) beserta aliran data dengan simbol-simbol standar yang mudah dipahami. Tujuan utama penggunaan flowchart adalah untuk menyederhanakan rangkaian proses atau prosedur untuk memudahkan pemahaman pengguna terhadap informasi tersebut. Untuk itu desain sebuah flowchart harus ringkas, jelas, dan logis[20].

### *2.2.8 Blackbox*

*Blackbox Testing* adalah pengujian perangkat lunak yang terfokus pada apakah unit program memenuhi kebutuhan yang disebutkan dalam spesifikasi. Pada blackbox testing, cara pengujian hanya dilakukan dengan menjalankan atau mengeksekusi unit atau modul, kemudian diamati apakah hasil dari unit itu sesuai dengan proses bisnis yang diinginkan. Jika ada unit yang tidak sesuai outputnya, maka untuk menyelesaikannya, diteruskan pada pengujian yang kedua, yaitu *white box testing*[21].