# BAB II
# TINJAUAN PUSTAKA

## II.1 Extension Browser

*Extension Browser* adalah sebuah aplikasi tambahan yang digunakan untuk menambah fungsionalitas atau dapat dikatan memperluas fungsionalitas yang dimiliki oleh sebuah *web browser*. Beberapa *web browser* besar yang sudah menggunakan teknologi *extension browser* adalah Google Chrome, Mozilla Firefox, dan Internet Explorer. Istilah lain yang digunakan untuk menyebut *extension browser* adalah Add-on dan Plug-in. Untuk membangun perangkat lunak *extension browser* ini data digunakan beberapa bahasa pemrograman diantaranya HTML, JavaScript, dan CSS [6].

## II.2 User Generated Content

UGC (*User Generated Content*) adalah informasi atau berbagai tipe konten baik berupa foto, video, dan lain-lain yang dipublikasikan kepada masyarakat berasal dari pengguna. Artinya informasi-informasi yang ada pada perangkat lunak seperti website ataupun aplikasi *mobile* itu tidak selalu berasal dari pendiri aplikasi tapi juga berasal dari penggunanya. Contoh aplikasi yang menerapkan konsep *user generated content* adalah Facebook, Twitter, Youtube dan Instagram.

Konsep *user generated content* pada dasarnya telah banyak mengubah cara berkomunikasi dengan internet misalnya pada bidang periklanan. Dengan menerapkan konsep tersebut khususnya pada jejaring sosial memiliki potensi yang besar dalam memasarkan produk yang dimiliki kepada masyarakat [7].

## II.3 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. JSON sendiri memiliki struktur data yang terdiri dari json array dan json object. Berikut ini adalah penjelasan dari kedua bagian tersebut:

4.1. JSON Array

JSON Array adalah kumpulan nilai yang terurutkan. Array dimulai dengan kurung kotak buka ([) dan diakhiri dengan kurung kotak tutup (]). Setiap nilai/*value* dipisahkan oleh tanda koma (,).

4.2. JSON Object

JSON Object adalah adalah kumpulan pasangan dari *string* dan *value*. Objek dimulai dengan kurung kurawal buka ({) dan diakhiri dengan kurung kurawal tutup (}). Setiap pasangan *string* dan *value* diikuti oleh titik dua (:) dan diakhir setiap pasangan dipisahkan oleh tanda koma (,). [8].

## II.4 Local Storage

*Local Stogare* atau penyimpanan local adalah sebuah tempat penyimpanan data sementara berupa *string* yang ada pada aplikasi HTML 5. Sebelum HTML 5 muncul, data disimpan ke dalam *cookies* termasuk juga setiap data *request* ke *server*. *Local Storage* sendiri sudah didukung oleh banyak *web browser* seperti Google Chrome, Internet Explorer, Mozilla Firefox, Safari, dan Opera [9].

## II.5 Object Oriented Analysis and Design

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 memerika requirement (syarat/keperluan) yang harus dipenuhi sebuah sistem) dari sudut pandang kelas-kelas dan objek-objek yang ditemui dalam ruang lingkup sistem. Sedangkan OOD adalah metode untuk mengarahkan arsitektur software yang didasarkan pada manipulasi objek-objek sistem atau subsistem [10].

### II.5.1 UML (Unified Modeling Language)

UML (*Unified Modeling Language*) adalah sebuah bahasa yang digunakan untuk memvisualkan, menspesifikasikan, membangun, dan merupakan

dokumentasi dari sebuah sistem pengembangan perangkat lunak berbasis objek. Di dalam UML terdiri dari beberapa diagram, yaitu:

1. Diagram Usecase

Diagram *usecase* menggambarkan apa saja aktifitas yang dilakukan oleh suatu sistem dari sudut pandang pengamatan luar. Yang menjadi persoalan adalah apa yang dilakukan bukan bagaimana melakukannya. Diagram *usecase* dekat kaitannya dengan kejadian-kejadian. Kejadian (skenario) merupaka contoh apa yang terjadi ketika seseorang berinteraksi dengan sistem.

2. Diagram Activity

Diagram *activity* berfokus pada aktifitas-aktifitas yang terjadi dan saling terkait dalam suatu proses tunggal. Dengan kata lain, diagram *activity* menjelaskan alur proses dari suatu proses sampai dengan selesai.

3. Diagram Class

Diagram *class* memberikan pandangan secara luas dari suatu sistem dengan menunjukan kelas-kelas dan hubungannya. Diagram *class* bersifat statis artinya menggambarkan hubungan apa yang terjadi bukan apa yang terjadi jika mereka berhubungan. Diagram *class* mempunyai tigas jenis hubungan, diantaranya:

a. Association

Hubungan antara bagian dari dua kelas. Terjadi *association* antara dua kelas jika salah satu bagian dari kelas mengetahui yang lainnya dalam melakukan suatu kegiatan.

b. Aggregation

*Association* dimana salah satu kelasnya merupakan bagian dari suatu kumpulan. *Aggregation* memiliki titik pusat yang mencangkup keseluruhan bagian. Misalnya orderDetail merupakan kumpulan dari order.

c. Generalization

Hubungan turunan dengan mengansumsikan satu kelas merupakan *superclass* dari kelas lain. *Generalizaion* memiliki tingkatan yang berpusat pada *superclass*. Misalnya payment adalah *superclass* dari cash, check, dan credit.

4. Diagram Sequence

Diagram *sequence* merupakan salah satu diagram interaksi yang menjelaskan bagaimana suatu operasi itu dilakukan, pesan apa yang dikirim, dan kapan pelaksanaannya. Diagram ini diatur berdasarkan waktu. Objek-objek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang terurut. *Lifetime* adalah garis dot (putus-putus) vertical yang menerangkan waktu terjadinya suatu objek. Setiap panan yang ada adalah pemanggilan suatu pesan. Panah berasal dari pengirim ke bagian paling atas dari batang kegiatan (*activation bar*) dari suatu pesan pada *lifetime* penerima. *Activation bar* menerangkan lamanya suatu pesan diproses.

## II.6 Pengujian Black Box

Pengujian black-box berfokus pada persyaratan fungsional perangkat lunak. Pengujian ini memungkinkan analis sistem memperoleh kumpulan kondisi input yang akan mengerjakan seluruh keperluan fungsional program. Tujuan dilakukannya pengujian black box adalah sebagai berikut:

1. Untuk mengetahui fungsi yang salah atau hilang
2. Untuk mengetahui kesalahan pada interface
3. Untuk mengetahui kesalahan pada struktur data atau akses database
4. Untuk mengetahui kesalahan performansi
5. Untuk mengetahui inisialisasi dan tujuan akhir

### II.6.1 Equivalence Partitioning

*Equivalence Partitioning* adalah metode pengujian *black-box* yang memecah atau membagi domain input dari program ke dalam kelas-kelas data sehingga test case dapet diperoleh. Equivalence Partioning berusaha untuk mendefinisikan kasus uji yang menemukan sejumlah jenis kesalahan dan mengurangi jumlah kasus uji

yang harus dibuat. Kasus uji yang didesain untuk *equivalence portioning* berdasarkan pada evaluasi dari ekuivalensi jenis atau *class* untuk kondisi input [11].

### II.6.2 Boundary Value Analysis

*Boundary Value Analysis* (BVA) mengarahkan pada pemilihan kasus uji yang melatih nilai-nilai batas. BVA merupakan desain teknik kasus uji yang melengkapi *equivalence partitioning*. Dari pada memfokuskan hanya pada kondisi *input*, BVA juga menghasilkan kasus uji dari domain *output*.

Panduan untuk BVA hampir sama pada beberapa bagian seperti yang disediakan untuk *equivalence partitioning*:

1. Jika kondisi *input* menspesifikasikan kisaran yang dibatasi oleh nilai a dan b, kasus uji harus dibuat dengan nilai a dan bm sedikit diatas dan sedikit dibawah a dan b.
2. Jika kondisi *input* menspesifikasikan sejumlah nilai, kasus uji harus dibuat dengan melatih nilai maksimum dan minimum, juga nilai-nilai sedikit di atas dan sedikit di bawah nilai maksimum dan minimum tersebut.
3. Aplikasikan panduan 1 dan 2 untuk kondisi *output*. Sebagai contoh, asumsikan tabel temperatur VS tabel tekanan sebagai *output* dari program analisis *engineering*. Kasus uji harus didesain untuk membuat laporan tekanan sebagai *output* yang menghasilkan nilai maksimum dan minimum yang mungkin untuk tabel masukan.
4. Jika struktur data program internal telah mendeskripsikan batasan (missal : *array* ditetapkan maks 100), maka desain kasus uji yang akan melatih struktur data pada batasan tersebut.