BAB II

TINJAUAN PUSTAKA

II.1. Pengertian Aplikasi

Program aplikasi adalah program siap pakai atau program yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain. Aplikasi juga diartikan sebagai penggunaan atau penerapan suatu konsep yang menjadi pokok pembahasan atau sebagai program komputer yang dibuat untuk menolong manusia dalam melaksanakan tugas tertentu. Aplikasi software yang dirancang untuk penggunaan praktisi khusus, klasifikasi luas ini dapat dibagi menjadi 2 (dua) yaitu:

a. Aplikasi software spesialis, program dengan dokumentasi tergabung yang dirancang untuk menjalankan tugas tertentu.
b. Aplikasi paket, suatu program dengan dokumentasi tergabung yang dirancang untuk jenis masalah tertentu (Rahmatillah ; 2011 : 3).

II.2. Latency

Network latency adalah istilah yang digunakan untuk menujukan jenis keterlambatan yang terjadi dalam komunikasi data melalui jaringan. Koneksi jaringan dimana penundaan kecil terjadi tersebut network low-latency (Jaringan Latency Rendah) sedangkan koneksi jaringan yang menderita penundaan yang lama disebut high-latency network (jaringan tinggi latency).

Latency tinggi menciptakan kemacetan dalam jaringan komunikasi. Ini mencegah data dari mengambil keuntungan penuh dari pipa jaringan dan efektif mengurangi bandwidth komunikasi. Dampak latency pada bandwidth jaringan dapat bersifat sementara atau persisten berdasarkan sumber penundaan (Adharul Muttaqin ; 2015 : 18).

II.3. Pengertian Transmisi Data

Transmisi data merupakan proses untuk melakukan pengiriman data dari salah satu sumber data ke penerima data menggunakan komputer / media elektronik. Untuk mengetahui lebih jauh tentang transmisi data beserta proses dan langkah kerjanya. Berikut ini merupakan beberapa hal yang berkaitan dengan proses ini:

II.3.1. Media Transmisi Data

Untuk melakukan transmisi data diperlukanlah suatu media, media ini sendiri memiliki beberapa macam seperti bus, kabel yang biasa terdapat pada perangkat internal komputer, sedangkan untuk eksternal komputer dalam transmisi data dapat menggunakan kabel eksternal (Wired) serta Wi-Fi (*Wireless*/Nirkabel).

Kabel (Wired)

Kabel / wired yang biasa digunakan untuk melakukan proses transmisi data terdapat beberapa macam yang diantaranya adalah sebagai berikut:

a. Kabel pilin: UTP Wired atau yang biasa dikenal dengan Unshielded Twisted Pair, kabel ini biasa digunakan untuk melakukan transmisi melalui jaringan komputer seperti di kantor-kantor / warnet-warnet. Selain UTP, STP (Shielded Twisted Pair) yang didalamnya terdapat beberapa kawat dalam satu bendel juga dapat digunakan untuk melakukan transmisi data.

b. Koaksial (coaxial cable): Kabel ini terdiri dari dua macam konduktor yang dipisahkan dengan menggunakan isolator.

c. Serat optik: Kabel ini biasa disebut dengan (fiber optic), dimana kabel yang dapat mengirimkan informasi dengan cara menghantarkan informasi / data menggunakan gelombang cahaya.

II.3.2. Jalur Transmisi Data

Jalur transmisi merupakan suatu alat yang mampu mengirimkan informasi dengan menggunakan peralatan yang lain. Jalur transmisi data ini dibagi menjadi 3 macam yakni Multicast, Broadcast dan Unicast.

a. **Multicast** Adalah suatu proses komunikasi terjadi melalui satu alat dengan alat lainnya. Dalam proses ini masing-masing alat / media yang terhubung dapat berkomunikasi menggunakan alat yang menghubunginya. Contohnya adalah server yang digunakan untuk mengakses internet. Server tersebut mampu melayani beberapa komputer yang terhubung dengan media, dan dalam proses ini komputer yang dihubungi mampu memberikan respon balik terhadap server tersebut.

b. **Broadcast** Adalah proses dalam pengiriman data atau informasi dari satu alat ke alat-alat lainnya. Dalam proses ini alat yang menerima informasi tidak bisa memberikan respon balik terhadap alat pengirim data / informasi. Beberapa contoh yang menggunakan jalur transmisi Broadcast adalah pemancar radio, pemancar televisi serta mengirim email menggunakan mailing list.

c. **Unicast** merupakan kontak informasi yang terjadi pada suatu alat dengan satu alat yang lain. Misalnya sewaktu menggunakan telepon, ketika salah satu telepon digunakan untuk menghubungi salah satu telepon yang lain, maka selain kedua telepon yang berhubungan

tersebut tidak dapat menghubungi salah satu dari telepon yang sedang terkoneksi / terhubung tersebut (Adharul Muttaqin ; 2015 : 18).

II.4. Websocket

Menurut jurnal yang ditulis oleh Sharma (2011, 50) ,Websocket di HTML5 merupakan kemajuan di bidang komunikasi HTTP. Spesifikasi dari Websocket memungkinkan saluran komunikasi dua arah *single-socket* untuk mengirim dan menerima informasi antara *browser* dan *server*. Dengan demikian, Websocket menghindari koneksi dan portabilitas masalah teknik lainnya dan memberikan solusi yang lebih efisien daripada polling Ajax. Saat ini Websocket di HTML5 adalah sarana terdepan untuk memfasilitasi *full-duplex,* pertukaran data di web secara *real time*. Websocket menyediakan lintasan sederhana dari *firewall* dan router dan kompatibel dengan data biner. Websocket juga memungkinkan pertukaran data dengan *cookie-based authentication.*

Tabel II.1. Perbandingan Websocket Dengan Teknologi Lain

Parameters Techniques	Cross-browser compatible	Cross-site browser security enforeed	Capability to receive HTTP status codes (error check)	Support for HTTP GET and POST	Compatibility in HTML Version	Plug-ins Requirement	Cookies Access	Capability to identily Data origin
XDM [8,9]	No	Yes	Yes	Yes	HTML 4	No	No	Yes
Window, name property [10,11]	Yes	No	No	Yes	HTML 4	No	Yes	No
iFrame [6,12,13,14]	Yes	No	No	Yes	HTML 4	No	No	No
Flash Plug-in [17]	Yes	Yes	Yes	Yes	HTML 4	Yes	Own Cookies	Yes
Document.domain proxy [10,13,18]	Yes	No	Yes	Yes	HTML 4	No	Yes	Yes
Microsoft Silverlight [19]	Yes	Yes	Yes	Yes	HTML 4	Yes	Yes	Yes
JSONP [20,21]	Yes (Server dependent)	No	No	No	HTML 4	No	No	Yes

JSONRequest [22]	Yes	Yes	Yes	Yes	HTML 4	No	No	Yes
Fragment Identifier [23,24]	Yes	No	Yes	Yes	HTML 4	No	No	Yes
Google Gears [6,25]	Yes	No	Yes	Yes	HTML 4	No	Yes	Yes
XDR [21,26]	No (e,I,8,ff,3,5)	Yes	Yes	Yes	HTML 4	No	No	Yes
CSSHttpRequest [27]	Yes	No	No	No	HTML 4	No	No	No
Mozila Signed Scripts [28]	Yes	Yes	Yes	Yes	HTML 4	No	Yes	Yes

(Sumber : Sharma ; 2010 : 50)

Tabel diatas membandingkan parameter yang berbeda untuk setiap teknologi yang berbeda yang digunakan dalam implementasi komunikasi *cross-site* pada *server* QIIIP. Empat belas teknologi dibandingkan terhadap berbagai parameter seperti c*ross-browser compatible, cross-site browser security enforcement, http status codes*, dll.

II.5. Algoritma RC4

Algoritma RC4 merupakan stream cipher yang didesain oleh Rivest untuk RSA Data Security (sekarang RSA Security) pada 1987. RC4 menggunakan panjang variabel kunci dari 1 s.d 256 byte untuk menginisialisasi state tabel. State table digunakan untuk pengurutan menghasilkan byte pseudorandom yang kemudian menjadi stream pseudo-random. Setelah di-XOR dengan plaintext sehingga didapatkan ciphertext. Tiap elemen pada state table di swap sedikitnya sekali. Kunci RC4 sering dibatasi sampai 40 bit, tetapi dimungkinkan untuk mengunakan kunci 128 bit. RC4 memiliki kemampuan penggunaan kunci antara 1 sampai 2048 bit (Yuri Ariyanto ; 2013 : 2).

II.5.1. Algoritma RC6

Algoritma RC6 adalah suatu algoritma kriptografi block cipher yang dirancang oleh Ronald L. Rivest, Matt J.B. Robshaw, Ray Sidney, dan Yuqin Lisa Yin dari RSA Laboratories. Algoritma ini pada mulanya dirancang untuk menjadi AES (Advance Encryption Standard). Algoritma RC6 ini berhasil menjadi finalis dan menjadi kandidat kuat untuk menjadi AES walaupun pada akhirnya algoritma ini tidak terpilih menjadi AES melainkan algoritma rinjdael. Versi 1.1 dari RC6 mulai dipublikasikan pada tahun 1998. Dasar desain dari algoritma RC6 ini didasarkan pada pendahulunya yaitu algoritma RC5. Algoritma RC6 merupakan algoritma dengan parameter penuh, algoritma RC6 dispesifikasikan dengan notasi RC6-w/r/b. Dimana w adalah ukuran dari word dalam bit, karena pada RC6 menggunakan 4 buah register maka word adalah ukuran blok dibagi 4. r adalah jumlah iterasi, dimana r tidak boleh negatif. Dan b adalah panjang kunci dalam bytes. Pembentukan kunci internal yang akan digunakan pada proses enkripsi dan dekripsi dari algoritma RC6 menggunakan pembentukan kunci internal dari algoritma RC6. Proses untuk membangun kunci-kunci internal menggunakan dua buah konstanta yang disebut dengan “magic constant” (Rangga Wisnu Adi Permana : 2).

II.6. JavaScript

Javascript bukan bahasa berorientasi objek, melainkan bahasa berbasis objek. Bahasa berorientasi objek harus mendukung tiga konsep dasar, yaitu pengkapsulan (*encapsulation*), perwarisan (*inheritance*) dan polimorfisme (*polymorphism*). Javascript hanya mendukung pengkapsulan, itupun tidak 100% benar. Program *JavaScript* dituliskan pada file *HTML* (*.html* atau *.htm*) dengan menggunakan tag container <*SCRIPT*>. Dengan kata lain, anda tidak perlu menuliskan program *JavaScript* pada file terpisah (meskipun anda bisa juga

melakukannya). Ingat bahwa yang dimaksud dengan tag container adalah tag yang diawali dengan <NAMA_TAG> dan diakhiri dengan </NAMA_TAG>. Beberapa contoh tag container adalah <HTML></HTML>, <HEAD></Body>, dsb (Antony Pranata ; 2001 : 11).

II.7. UML (*Unified Modeling Language*)

Menurut Windu Gata (2013 : 4) Hasil pemodelan pada OOAD terdokumentasikan dalam bentuk *Unified Modeling Language*(UML). UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak.

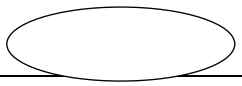
UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem.

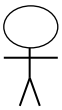
Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

1. *Usecase* Diagram

   *Usecase*diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Usecase* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *usecase* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *usecase* diagram, yaitu :

Tabel II.2. Simbol *UseCase*

Gambar	Keterangan
	*Usecase* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan pesan antar unit dengan aktor, biasanya dinyatakan

	dengan menggunakan kata kerja di awal nama *usecase*.
	Aktor adalah *abstraction*dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *usecase*, tetapi tidak memiliki control terhadap *usecase*.
	Asosiasi antara aktor dan *usecase*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.
	Asosiasi antara aktor dan *usecase* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
	Include, merupakan di dalam *usecase* lain (*required*) atau pemanggilan *usecase* oleh *usecase* lain, contohnya adalah pemanggilan sebuah fungsi program.
	Extend, merupakan perluasan dari *usecase* lain jika kondisi atau syarat terpenuhi.

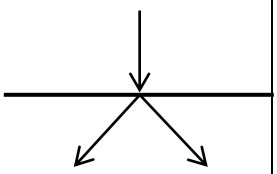
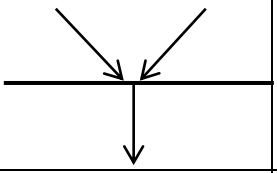
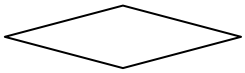
(Sumber : WinduGata ; 2013 : 4)

2. Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram,* yaitu :

Tabel II.3. Simbol *Activity Diagram*

Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*Endpoint*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.

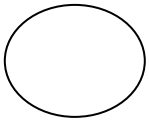
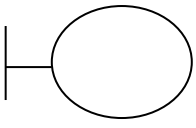
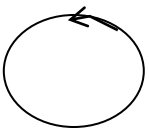
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points,* menggambarkan pilihan untuk pengambilan keputusan, *true*, *false.*
	*Swimlane,* pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

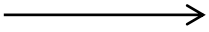
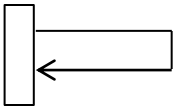
(Sumber : WinduGata ; 2013 : 6)

3. Diagram Urutan (*Sequence Diagram*)

Sequence diagram menggambarkan kelakuan objek pada *usecase* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram*, yaitu :

Tabel II.4. Simbol *Sequence Diagram*

Gambar	Keterangan
	*EntityClass,* merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*BoundaryClass,* berisi kumpulan kelas yang menjadi *interface*atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan *form*cetak.
	*Control class,* suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.

	*Message*, simbol mengirim pesan antar *class*.
	*Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	*Activation, activation*mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline*terdapat *activation*.

(Sumber : WinduGata ; 2013 : 7)

4. *Class Diagram* (Diagram Kelas)

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggng jawab entitas yang menentukan perilaku sistem.

Class diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*), *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti.

Tabel II.5. *MultiplicityClass Diagram*

Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih

0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber : WinduGata ; 2013 : 9)