BAB II

LANDASAN TEORI

II.1. Metode *Equrectangular Approximation*

Proyeksi *Equirectangular Approximation* atau sering disebut sebagai equidistant cylindrical projection, geographic projection, atau la carte parallélogrammatique projection, adalah proyeksi sederhana pada peta. Proyeksi peta menggunakan teknik ini diukur dari meridian peta terhadap garis tegak lurus secara vertikal dan dalam ruang yang konstan. Pada proyeksi ini, luas pada permukaan peta tidak selalu sama yang disebabkan oleh distorsi atas bentuk bumi yang sebenarnya tidak benar-benar bulat. Proyeksi peta dan pengukuran jarak menggunakan formula ini digunakan sebagai standar proyeksi pada dataset global seperti Cartesia dan NASA World Wind karena keterhubungan antara piksel pada gambar peta dan posisi lokasi geografis bumi cukup sederhana.

Pengukuran jarak menggunakan *Equirectangular Approximation* tepat digunakan jika perangkat komputer yang digunakan tidak memiliki performance yang memadai dan akurasi tidak terlalu penting serta jarak yang diukur masih dalam skala kecil, Teorema pitagoras dapat digunakan pada proyeksi *Equirectangular Approximation* menggunakan formula di bawah ini:

$$x = \Delta\lambda . \cos(\varphi)$$

$$y = \Delta\varphi$$

$$d = R\sqrt{(x^2 + y^2)}$$

Dimana:

λ: titik koordinat longitudinal

φ: titik koordinat latitude

x: posisi horizontal pada peta

y: posisi vertikal pada peta

d: jarak antara kedua posisi (Salah M El – Sayed : 2014 ; 57).

II.2. Rute

Pencarian rute terpendek adalah menentukan jalur yang paling optimal,yaitu jalur dengan rute terpendek dan biaya terkecil dalam penerapannya dapat dilakukan pada satu atau lebih asal pencarian rute ke satu atau lebih tujuan melalui jaringan yang terhubung. Dalam beberapa aplikasi, juga bermanfaat untuk mengetahui jalur terpendek dua atau tiga alternatif tambahan misalnya, dalam rangka meningkatkan efektivitas pemberian informasi perjalanan, kebutuhan untuk memberikan beberapa jalur alternatif bagi pengguna jalan dalam mengemudi. Dalam penentuan rute dapat memanfaatkan proses secara metaheuristic yaitu metoda untuk mencari solusi yang memadukan interaksi antara prosedur pencarian local dan strategi yang lebih tinggi untuk menciptakan proses yang mampu keluar dari titik-titik local optimal dan melakukan pencarian di ruang solusi untuk menemukan solusi global (Ivana Varita ; 2013 : 185)

II.3. Studio Musik

Pengertian umum studio musik adalah ruang di mana musisi bermain musik, suara direkam, dandiedit. Secara umum studio musik memiliki dua bagian, yaitu take room di mana suara musisi dan vokalis dimainkan dan ditangkap oleh mikrofon. Kemudian, ruang kontrol adalah tempat di mana seorang sound engineer merekam permainan musisi dan vokalis yang kemudian disimpan, diedit, dipoles, di-mix, diputar ulang, dan diproses mastering.

Ada beragam jenis ukuran studio musik, mulai dari studio mini dengan kapasitas tidak lebih dari delapan orang sampai dengan studio besar yang dapat menampung seluruh musisi okestra simfoni besar. Dalam mini home studio, mungkin hanya ada mix board dan perangkat elektronik lainnya dalam satu ruangan bersama musisi dan sound engineer. Dalam fasilitas studio yang sedang dan dengan fungsi ruangan terpisah, pada gilirannya akan terbagi antara recording studio dan ruang kontrol (Dwi Yudha ; 2013 : 1).

II.4. *PHP*

PHP adalah sebuah bahasa pemrograman scripting untuk membuat halaman web yang dinamis. Walaupun dikenal sebagai bahasa untuk membuat halaman web, tapi *PHP* sebenarnya juga dapat digunakan untuk membuat aplikasi command line dan juga GUI. *Website* yang dibuat menggunakan *PHP* memerlukan *software* bernama *webserver* tempat pemrosesan kode *PHP* dilakukan. Server web yang memiliki *software PHP* parser akan memproses input berupa kode *PHP* dan menghasilkan output berupa halaman web. *PHP* bersifat

terbuka dan multiplatform, karenanya dapat dijalankan di banyak merek web server (seperti apache dan IIS) (Ali Zaki ; 2015 : 2)

II.5. *MySQL*

MySQL merupakan salah satu sistem *database* yang sangat handal karena menggunakan sistem *SQL*. Pada awalnya *SQL* berfungsi sebagai bahasa penghubung antara program *database* dengan bahasa pemrograman yang kita gunakan. Dengan adanya *SQL* maka para pemrogram jaringan dan aplikasi tidak mengalami kesulitan sama sekali di dalam menghubungkan aplikasi yang mereka buat. Setelah itu *SQL* dikembangkan lagi menjadi sistem *database* dengan munculnya *MySQL*. *MySQL* merupakan *database* yang sangat cepat, beberapa user dapat menggunakan secara bersamaan dan lebih lengkap dari *SQL*. *MySQL* merupakan salah satu *software* gratis yang dapat di-download melalui situsnya. *MySQL* merupakan sistem manajemen *database*, relasional sistem *database* dan *software open source*. (Stendy B. Sakur ; 2015 : 58)

II.6. 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

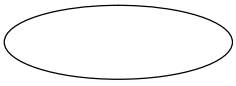
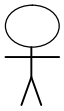
saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem.

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

1. *Use case* Diagram

   Use case diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *use case* 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 *use case* diagram, yaitu :

Tabel II.1. Simbol *Use Case*

Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	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 *use case*, tetapi tidak memiliki control terhadap *use case*.
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.

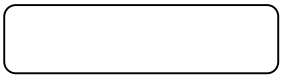
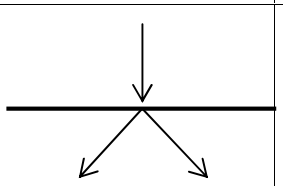
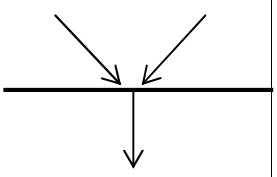
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
	Include, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	Extend, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

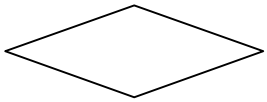
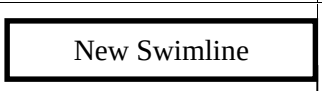
(Sumber : Windu Gata ; 2013 : 4)

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.2. Simbol *Activity Diagram*

Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, 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.

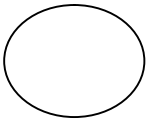
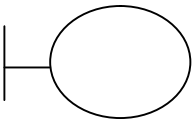
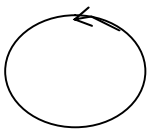
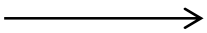
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true, false.*
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

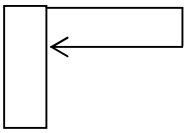
(Sumber : Windu Gata ; 2013 : 6)

3. Diagram Urutan (*Sequence Diagram*)

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

Tabel II.3. Simbol *Sequence Diagram*

Gambar	Keterangan
	*Entity Class*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*Boundary Class*, 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 : Windu Gata ; 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.4. ***Multiplicity Class 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 : Windu Gata ; 2013 : 9)