BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terkait

Untuk mendukung keberhasilan penelitian ini, penulis melakukan pendekatan teoritis melalui beberapa literatur yang berhubungan dengan penelitian yang dilakukan. Beberapa uraian penelitian terdahulu yang menjadi acuan dalam penelitian ini yaitu:

Canggih Ajika Pamungkas, 2019, dengan judul penelitian "Aplikasi Penghitung Jarak Koordinat Berdasarkan Latitude Dan Longitude Dengan Metode Euclidean distance Dan Metode Haversine". Penelitian ini menghasilkan Aplikasi dapat digunakan pada perangkat android b. Pengujian menunjukkan hasil sama saat perhitungan jarak antara metode Euclidean distance dan metode Haversine

Febriliyan Samopa, 2017, dengan judul penelitian "Penerapan Euclidean distance Pada Pencocokan Pola Untuk Konversi Citra Ke Teks". Penelitian ini menghasilkan Besar kecilnya nilai error tidak ditentukan oleh ukuran citra masukkan, akan tetapi ditentukan oleh intensitas dan pola dari citra masukan. Semakin besar nilai intensitas dari citra masukan, maka akan menghasilkan output yang hampir mendekati citra masukan. Jika insentitas citra masukan kecil, maka akan kesulitan dalam mendeteksi piksel obyek tersebut, dan akan dianggap putih sehingga pada hasil outputnya akan banyak informasi yang hilang. Hal ini mengakibatkan citra masukan tidak dan hasil output tidak sesuai.

M. Nishom, 2019, dengan judul penelitian "Perbandingan Akurasi

Euclidean Distance, Minkowski Distance, dan Manhattan Distance pada Algoritma KMeans Clustering berbasis Chi-Square". Penelitian ini menghasilkan Perbandingan akurasi metode pengukuran jarak (euclidean, manhattan, dan minkowski) untuk pelabelan klaster status disparitas kebutuhan Guu telah dilakukan dan memberikan nilai atau tingkat akurasi yang tinggi, yaitu 84.47% (untuk metode euclidean distance), 83.85% (untuk metode manhattan distance), dan 83.85% (untuk metode minkowski). Dengan demikian, dapat disimpulkan bahwa metode euclidean meupakan metode terbaik untuk diterapkan dalam algoritma KMeans Clustering.

Evi Yulianti dkk, 2020, dengan judul penelitian "Aplikasi Pencarian Rute Terpendek Lokasi Kuliner Khas Palembang Menggunakan Algoritma Euclidean Distance dan A*(Star)". Penelitian ini menghasilkan penelitian ini maka disimpulkan bahwa Algoritma Euclidean Distance dan A* (Star) dapat digunakan untuk melakukan pencarian rute terpendek lokasi kuliner yang ada di kota palembang. Berdasarkan hasil akurasi yang dicapai pada pengujian menunjukkan bahwa Nilai MAPE (mean absolute percentage error) untuk evaluasi prediksi akurasinya yaitu 4,4%. Pengujian yang dilakukan menghasilkan kesimpulan bahwa metode yang dilakukan memiliki tingkat akurasi tinggi. Peneliti menyarankan untuk penelitian selanjutnya dapat mengembangkan aplikasi ini seperti dengan menambahan fitur seperti suara untuk mengarahkan ke tempat lokasi tujuan dan juga menghitung kemacetan lalu lintas jalan

II.2. Landasan Teori

II.2.1. Euclidean distance

Euclidean distance adalah perhitungan jarak dari 2 buah titik dalam Euclidean space. Euclidean space diperkenalkan oleh seorang matematikawan dari Yunani sekitar tahun 300 B.C.E. untuk mempelajari hubungan antara sudut dan jarak. Euclidean ini biasanya diterapkan pada 2 dimensi dan 3 dimensi. Tapi juga sederhana jika diterapkan pada dimensi yang lebih tinggi. Teknik cross validasi digunakan untuk mencari nilai k yang optimal dalam mencari parameter terbaik dalam sebuah model. Jarak Euclidean menurut McAndrew digunakan untuk menghitung jarak antara dua vektor yang berfungsi menguji ukuran yang bisa digunakan sebagai interpretasi kedekatan jarak antara dua obyek yang direpresentasikan dalam persamaan.

II.2.2 Android SDK

(Software Development Kit) Menurut Wandy Damarullah (2018) Android SDK merupakan tools bagi para programmer yang ingin mengembangkan aplikasi berbasis google android. Android SDK mencakup seperangkat alat pengembangan yang komprehensif. Android SDK terdiri dari debugger, libraries, handset emulator dokumentasi, contoh kode, dan tutorial.

II. 2.3. Java Development Kit (JDK)

Menurut Andi Juansyah (2015) Java Development Kit (JDK) adalah sekumpulan perangkat lunak yang dapat kamu gunakan untuk mengembangkan perangkat lunak yang berbasis Java, sedangkan JRE adalah sebuah implementasi

dari Java Virtual Machine yang benarbenar digunakan untuk menjalankan program java. Baisanya, setiap JDK berisi satu atau lebih JRE dan berbagai alat pengembangan lain seperti sumber compiler java, bundling, debuggers, development libraries dan lain sebagainya.

II.2.4. Sistem Operasi Android

Menurut Heru Supriyono, dkk (2014) Android adalah salah satu platform sistem operasi yang digemari masyakat karena sifatnya yang open source sehingga memungkinkan pengguna untuk melakukan pengembangan. Android merupakan generasi baru platform mobile berbasis linux yang mencakup sistem operasi, middleware, dan aplikasi. Arsitektur Android terdiri dari bagian-bagian seperti berikut:

Applications dan Widgets: layer (lapisan) dimana pengguna hanya berhubungan dengan aplikasi saja.

1. Applications Framework: lapisan dimana para pengembang melakukan pembuatan aplikasi yang akan dijalankan di sistem operasi Android dengan 22 komponen-komponennya meliputi views, contents provider, resource manager, notification manager, activity manager.
2. Libraries: lapisan dimana fitur-fitur android berada yang berada diatas kernel meliputi library C/C++ inti seperti Libc dan SSL.
3. Android Run Time: lapisan yang membuat aplikasi Android dapat dijalankan dimana dalam prosesnya menggunakan implementasi Linux yang terbagi menjadi dua bagian yaitu Core Libraries dan Dalvik virtual Machine.

4. Linux Kernel: Layer yang berisi file-file system untuk mengatur processing, memory, resource, driver, dan sistem operasi android lainnya. Sistem operasi yang mendasari Android dilisensikan dibawah GNU, GPLv2 (General Public License verse 2) yang sering dikenal dengan istilah copyleft. Pendistribusian Android dibawah lisensi dari Apache Software yang memungkinkan untuk distribusi kedua dan seterusnya.

II.2.5. UML (Unified Modeling Language)

UML (*Unified Modeling Language*) digunakan sebagai suatu cara untuk mengkomunikasikan idenya kepada para pemrogram serta calon pengguna sistem perangkat lunak. Dengan adanya bahasa yang bersifat standar, komunikasi perancang dengan pemrogram (komunikasi antar anggota kelompok pengembang) serta calon pengguna diharapkan menjadi mulus. Adapun pengertian UML menurut para ahli dapat dipaparkan sebagai berikut:

Menurut Shofwan Hanief & Dian Pramana (2018 : 166) menyatakan bahwa*Unified Modelling Language* (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisas, merancang dan mendokumentasikan sistem piranti lunak.

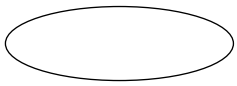
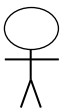
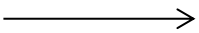
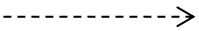
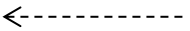
Menurut Eng RH. Sianipar (2016 :75) menyatakan bahwa UML merupakan singkatan *Unifed Modelling Language*, yang telah menjadi notasi popular untuk mempresentasikan perancangan atas sebuah program berorientasi objek.

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

1. *Use case* Diagram

Menurut Sri Mulyani (2016 : 245) menyatakan bahwa *Use Case Diagram* yaitu *diagram* yang menggambarkan dan merepresentasikan aktor, *use cas*e dan *dependencies* suatu proyek dimana tujuan dari diagram ini adalah menjelaskan konsep hubungan antara sistem dengan dunia luar. Jadi, dapat disimpulkan *use case* adalah langkah-langkah atau urutan kegiatan yang dilakukan aktor dan sistem informasi yang akan dibuat.Secara singkat, *use case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah sistem informasi dan siapa saja yang berhak menggunakan.

Tabel II.1. Simbol *Use Case*

Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar 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 kontrol 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 mengindikasikan aliran data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengindikasikan 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 : Sri Mulyani : 2016: 245)

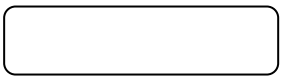
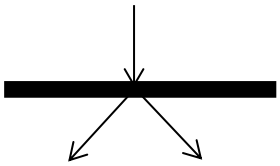
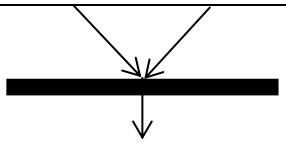
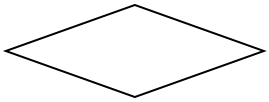
2. Diagram Aktivitas (*Activity Diagram*)

Activity diagram menggambarkan rangkaian aliran dari aktivitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti use case atau interaksi.

Menurut Sri Mulyani (2016 : 249) *activty diagram* adalah diagram UML yang digunakan untuk menggambarkan alur aktivitas dari suatu proses. Menurut Muhammad Muslihudin dan Oktafianto (2016 :63) mengungkapkan diagram aktivitas adalah tipe khusus dari diagram status yang memperlihatkan aliran dari suatu aktivitas ke aktivitas lainnya dalam suatu sistem.

Simbol-simbol yang digunakan dalam *activity diagram* dapat dilihat pada tabel II.2:

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 paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu.
	*Join* (penggabungan) atau *rake*, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true, false.*

New Swimline	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

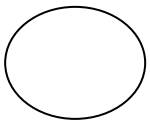
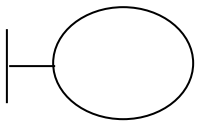
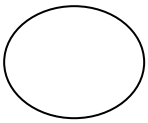
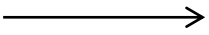
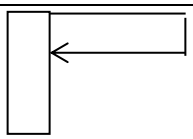
(Sumber : Sri Mulyani : 2016: 249)

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.

Menurut (Irmayani & Susyatih, 2017), *Sequence Diagram* menggambarkan bagaimana sistem merespon kegiatan *user*. Simbol-simbol yang digunakan dalam *sequence diagram* dapat dilihat pada tabel II.3:

Tabel II.3. 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.
	*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 sistem 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 : Irmayani & Susyati, 2017)

4. Diagram Kelas (*Class Diagram)*

   Diagram kelas (*Class Diagram*) adalah diagram yang digunakan untuk menampilkan beberapa kelas serta paket-paket yang ada dalam sistem atau perangkat lunak yang sedang kita kembangkan. Dan berikut ini merupakan penjelasan mengenai *class diagram*:

   Menurut Sri Mulyani (2016 : 247) menyatakan bahwa *class diagram* adalah diagram yang digunakan untuk mempresentasikan kelas, komponen-komponen kelas dan hubungan antara masing-masing kelas. Simbol *multiplicity class diagram* dapat dilihat pada Tabel II.4.:

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 : Sri Mulyani : 2016 : 247)