BAB II

TINJAUAN PUSTAKA

BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terkait

Pada penelitian terkait yang dilakukan oleh Dona Marcelina dan Evi Yulianti; (2020) dengan judul penelitian **Aplikasi Pencarian Rute Terpendek Lokasi Kuliner Khas Palembang Menggunakan *Algoritma Euclidean Distance* dan *A*(Star)***. Berdasarkan hasil yang didapat dalam 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.

Pada penelitian terkait yang di lakukan oleh Suparmi dan Soeheri; (2020) dengan judul penelitian **Sistem Informasi Geografis Pemetaan Tempat Kost Berbasis Web Menggunakan Metode *Euclidean Distance***. Berdasarkan penelitian yang telah dilakukan selama membuat aplikasi sistem informasi geografis ini, maka dapat ditarik beberapa kesimpulan sebagai berikut:

1. Membuat suatu sistem yang dapat memudahkan masyarakat dalam mencari informasi dan lokasi tempat kos daeah kampus.

2. Mewujudkan sebuah sistem untuk jarak terdekat tempat kos daerah kampus dengan Metode *Euclidean distance*.

Pada penelitian terkait yang di lakukan oleh Yusup Miftahuddin, dkk (2020) dengan judul penelitian **Perbandingan Metode Perhitungan Jarak Euclidean, Haversine, Dan Manhattan Dalam Penentuan Posisi Karyawan (STUDI KASUS : INSTITUT TEKNOLOGI NASIONAL BANDUNG).** Setelah melakukan serangkaian penelitian, maka kesimpulan yang diperoleh adalah Hasil dari setiap perhitungan jarak yang berbeda dapat disebabkan oleh konsep setiap metode perhitungan. Manhattan menghasilkan deviasi paling besar. Konsep perhitungan jarak manhattan yaitu menerapkan konsep pencarian selisih murni antar data. Hal tersebut kurang cocok terhadap perhitungan jarak yang menggunakan variabel koordinat latitude dan longitude.

Pada penelitian terkait yang di lakukan oleh Canggih Ajika Pamungkas; (2019) dengan judul penelitian **Aplikasi Penghitung Jarak Koordinat Berdasarkan Latitude dan Longitude dengan Metode *Euclidean Distance* dan Metode *Haversine***. Berdasarkan hasil dan pembahasan yang telah dilakukan dapat ditarik kesimpulan sebagai berikut:

a. Aplikasi dapat digunakan pada perangkat android.
b. Pengujian menunjukkan hasil sama saat perhitungan jarak antara metode *Euclidean Distance* dan metode *Haversine*.

Pada penelitian terkait yang dilakukan oleh Fitriyani, dkk; (2020) dengan judul penelitian **Penerapan Algoritma *Euclidean Distance* Untuk Pemilihan Paket Internet Berdasarkan Wilayah.** Berdasarkan perancangan, pembuatan,

uji coba dan analisa program aplikasi presensi ini, dapat diambil kesimpulan yaitu Aplikasi sistem pemilihan paket internet menggunakan algoritma Euclidean Distance dapat diterapkan untuk membantu konsumen khususnya konsumen BIP Ponsel Banjarbaru untuk menentukan paket internet yang sesuai untuk mereka gunakan.

II.2. Landasan Teori

II.2.1. Aplikasi

Aplikasi adalah satu unit perangkat lunak yang dibuat untuk membantu melayani kebutuhan seseorang untuk melakukan aktivitas. Dengan adanya aplikasi maka kebutuhan akan pelayanan sebuah aktivitas menjadi lebih baik. Dimana setiap pekerjaan dapat dilakukan dengan mudah kalau menggunakan sebuah aplikasi (Agusdi Syafrizal: 2018).

II.2.2. Peta

Pengertian peta secara umum adalah gambaran dari permukaan bumi yang digambar pada bidang datar, yang diperkecil dengan skala tertentu dan dilengkapi symbol sebagai penjelas. Peta Digital adalah kumpulan data yang mewakili informasi spasial dan atribut, disimpan di komputer. Ini adalah penyimpanan informasi spasial seperti gambar elektronik yang dibuat dari elemen grafis sederhana (garis, titik, lingkaran, dll.) yang disusun berlapis-lapis, dengan tujuan keluaran cetak atau layar (Yandi Nasution, dkk; 2022).

II.2.3. *Mapbox*

Mapbox adalah sebuah platform pemetaan open source yang bekerja dan merilis sebagai kode sebanyak mungkin. Sebagian besar data Mapbox menggunakan bantuan serta berinvestasi pada berbagai macam sumber data misalnya OpenStreetMap, USGS, Landsat, dan OpenAddresses. Mapbox mendukung berbagai macam aplikasi yang akan digunakan oleh penggunanya, baik mobile maupun online. Produk yang tersedia di Mapbox terdiri dari peta, satelit, server atlas, geocoding, dll. Mapbox mendukung beberapa aplikasi pengembang, diantaranya JavaScript, iOS, Android dan API. Mapbox telah digunakan untuk aplikasi Foursquare, Pinterest dan Evernote yang memudahkan penggunanya untuk menandai lokasi mereka kapanpun dan dimanapun. Pengguna Mapbox yang akan mendaftar disediakan berbagai pilihan akses data dengan berbagai pilihan biaya yang tentu saja mempengaruhi keberagaman fasilitas yang di dapat oleh pengguna (Windu P. Rimbing, dkk; 2017).

II.2.4. Objek Wisata

Objek wisata adalah fasilitas dasar yang harus di miliki dalam industri pariwisata. Hal ini menjadi tantangan dan sejumlah objek wisata yang kurang terawat, sehingga minat pengunjung berkurang (Dewi Mulyati, dkk; 2018).

II.2.5. *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. untuk mempelajari hubungan antara sudut dan jarak. *Euclidean* ini biasanya diterapkan pada 2 dimensi. kemudian juga bisa sederhana jika diterapkan pada dimensi lain yang lebih tinggi.

Metode *Euclidean Distance* merupakan suatu metode pencarian kedekatan jarak dari 2 buah variabel, selain mudah metode ini juga lebih efisien waktu, dan proses yang cepat. Jarak *Euclidean Distance* dapat dirumuskan sebagai berikut :

$$d = \sqrt{(x1 - x2) + (y1 - y2)} \quad (1)$$

dimana

d = jarak *euclidean distance* dalam derajat

x1= *latitude* lokasi awal

x2= *latitude* lokasi tujuan

y1=*longitude* lokasi awal

y2= *longitude* lokasi tujuan

Studi kasus Metode *Euclidean distance*

$d = \sqrt{(\varphi 1-\varphi 2)^2 + (\lambda 1-\lambda 2)^2} \cdot S$

S: Satuan derajat (1 derajat bumi = 111.319 km)

1 derajat (lintang/bujur) = 111.319 Km = 111319 meter

Koordinat peta terdiri dari titik (x,y)

dimana sumbu x adalah longitudinal

dan sumbu y adalah latitude.

φ (phi) lambang untuk latitude

λ (lambda) lambang untuk longitudinal.

Implementasi di program:

```
var S = 111.319;
var φ1 = lat1;
var λ1 = lon1;
var φ2 = lat2;
var λ2 = lon2;
var d = Math.sqrt(Math.pow(φ1-φ2,2) + Math.pow(λ1-λ2,2)) * S;
```

Untuk menenentukan jalur terdekat antara Universitas Muhammadiyah Sumatra Utara ke Kos Qu, Universitas Potensi Utama ke Kos Humairoh, Universitas Dharmawangsa Medan ke Rumah Samosir dengan menggunakan rumus Euclidean Distance dengan rute yang di tentukan, ekspresinya adalah sebagai berikut:

Universitas Muhammadiyah Sumatera Utara ke Kos Qu Perhitungan jarak koordinat antara titik koordinat (N 3.6144799, E 98.6755336º) 5ke koordinat (N. 3.617026, E 98.675750 º):

Euclidean[(3. 6144799, 98.6755336) → (3. 617026, 98.675750)]

$= \sqrt{(\varphi 1-\varphi 2)^2 + (\lambda 1-\lambda 2)^2} \cdot S$

$= \sqrt{(3.6144799-3.617026)^2 + (98.6755336-98.675750)^2} \times 111.319$

$= 0.284451177$

Universitas Dharmawangsa ke Runa Samosir Kos Perhitungan jarak koordinat antara titik koordinat (N. 3.6136471º, E 98.6734519 º) ke koordinat (N 3.613718º, E 98.673228 º):

Euclidean[(3.6136471, 98.6734519) → (3.613718, 98.673228)]

$= \sqrt{(\varphi 1-\varphi 2)^2 + (\lambda 1-\lambda 2)^2} \cdot S$

$= \sqrt{(3.\ 6136471\text{-}3.\ 613718)^2 + (98.6734519\text{-}98.673228)^2} \times 111.319$

$= 0.026144096$

Telah ditemukan bahwa dari kedua lokasi Universitas Muhammadiyah Sumatera Utara dan Kos Qu menggunakan metode Euclidean Distance jarak rute terdekat adalah Kos Qu lokasi Universitas Muhammadiyah Sumatera Utara dengan jarak terendah yaitu 0.284451177 Km [1].

II.2.6. *Android*

Android merupakan sistem operasi untuk perangkat *mobile* dengan berbasis *Linux*. *Android* menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi yang digunakan oleh bermacam perangkat *mobile*. *Android* dapat digunakan di *smartphone* dan juga tablet *PC*. Fungsinya sama seperti sistem operasi lainnya, seperti *Symbian* di Nokia, *iOS* di *Apple* dan *BlackBerry OS* (Canggih Ajika Pamungkas; 2019).

II.2.7. *Android Studio*

Android Studio merupakan salah satu penyedia *software* yang saat ini banyak digunakan para *developer* untuk membuat atau mengembangkan sebuah program seperti android *project*. *Software* ini dibuat oleh pihak *Google* yang berbasis *IntelliJ IDEA*. *Android Studio* dari tahun ke mengalami peningkatan keunggulan seperti dalam segi tampilan yang dapat *multi screen* sehingga *developer* lebih mudah dapat menulis kode. Selain itu, Android Studio juga mempunyai fitur *fleksible gradle* yang mana berfungsi untuk membuat banyak

APK sekaligus dengan fitur aneka macam meskipun menggunakan code yang sama dalam pembuatannya (Afrian Dwi Putra & Aslan Alwi; 2018).

Versi-versi *Android* sebagai berikut :

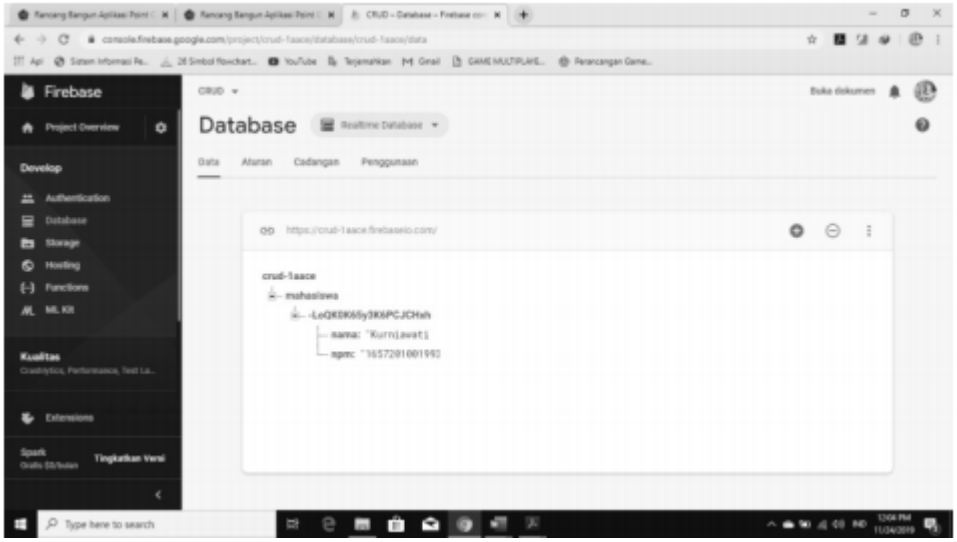
1. *Android* v1.0 *Astro* (*Alpha*)
2. *Android* v1.5 *Cupcake*
3. *Android* v1.6 *Donut*
4. *Android* v2.0-2.1 *Eclair*
5. *Android* v2.2 *Frozen Yogurt (Froyo)*
6. *Android* v2.3 *Gingerbread*
7. *Android* v3.0-3.2 *Honeycomb*
8. *Android* v4.0 *Ice Cream Sandwich*
9. *Android* v4.1-4.3 *Jelly Bean*
10. *Android* v4.4 *Kitkat*
11. *Android* v5.0-5.1 *Lollipop*
12. *Android* v6.0 *Marshmallow*
13. *Android* v7.0 *Nougat*
14. *Android* v8.0 *Oreo*

(Yoyon Efendi; April 2018)

II.2.8. *Firebase Realtime Database*

Firebase Realtime Database sendiri merupakan basis data berbasis *cloud NoSQL* yang menyinkronkan data di semua klien secara *realtime*, dan menyediakan fungsionalitas *offline*. Data yang diinputkan disimpan ke dalam

database Realtime sebagai *JSON*. Semua klien yang terhubung dan saling berbagi dalam satu waktu, secara otomatis akan menerima pembaruan dengan data terbaru



Gambar II.1. Contoh *Firebase Realtime Database*

Sumber : (Kurniawati dan Lukman Bacthiar ; 2020 : 59).

Database ini sendiri merupakan database yang disediakan oleh *Google* dan memiliki 2 solusi *database* dalam pengunaannya, Solusi pertama disebut *Cloud Firetstore. Cloud Firestore* adalah *database* terbaru dari *Firebase* untuk pengembangan aplikasi seluler. *Database* ini melanjutkan keberhasilan Realtime *Database* dengan model data baru yang lebih *intuitif* (Kurniawati dan Lukman Bacthiar ; 2020 : 59).

II.2.9. *UML*

Unified Modelling Language (*UML)* merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah

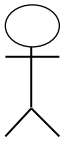
sistem *software* yang terkait dengan objek. *UML* merupakan salah satu alat bantu yang sangat handal dalam bidang pengembangan sistem berorientasi objek karena *UML* menyediakan bahasa pemodelan visual yang memungkinkan pengembang sistem membuat *blue print* atas visinya dalam bentuk yang baku. (Ommi Alfina, Fitriana Harahap ; 2019).

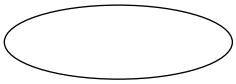
Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan *UML* adalah sebagai berikut :

1. *Use case* 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 diagram* dapat digambarkan dengan sumber-sumber pada tabel II.1.

Tabel II.1. Simbol *Use Case Diagram*

Gambar	Nama	Keterangan
	*Actor*	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan use case.
	*Depedency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (independent) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (independent).
	Generalization	Hubungan dimana objek anak

		(descendent) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (ancestor).
	*Include*	Menspesifikasikan bahwa use case sumber secara eksplisit.
	*Extend*	Menspesifikasikan bahwa use case target memperluas perilaku dari use case sumber pada suatu titik yang diberikan.
	Association	Apa yang menghubungkan antara objek satu dengan objek lainnya.
	*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
	*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor.
	*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
	*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

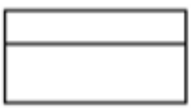
(Sumber : Adi Fitra Andikos, M.Kom; 2019)

2. *Class Diagram* (Diagram Kelas)

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (*atribut*/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (Fitriana Harahap; 2017).

Class diagram dapat digambarkan dengan simbol-simbol seperti pada tabel II.2.

Tabel II.2. Simbol *Class Diagram*

Gambar	Nama	Keterangan
	Generalization	Hubungan dimana objek anak (descendent) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (ancestor). Mandiri (independent) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (independent).
	*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
	*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
	*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor.
	*Realization*	Operasi yang benar-benar dilakukan

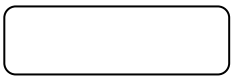
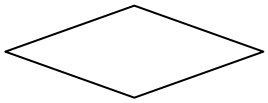
		oleh suatu objek.
	*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (independent) akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
	Association	Apa yang menghubungkan antara objek satu dengan objek lainnya.

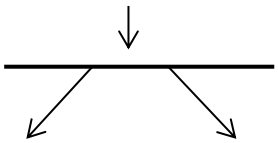
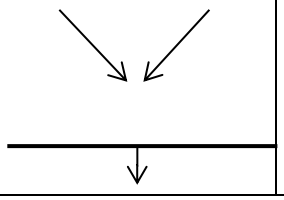
(Sumber : Adi Fitra Andikos, M.Kom; 2019)

3. Diagram Aktivitas (*Activity Diagram*)

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu di perhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem seperti pada tabel II.3.

Tabel II.3. Simbol Diagram Aktivitas

Gambar	**Nama**	**Keterangan**
	Status Awal	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
	Aktivitas	Aktivitas yang dilakukan oleh sistem, aktivitas biasanya diawali dengan kata kerja.
	*Decision*	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.

	*Join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
	Status Akhir	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
	*Swimlane*	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.
	*Fork,*	Digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel.
	*Join,*	Digunakan untuk menunjukkan kegiatan yang digabungkan.

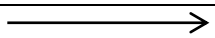
(Sumber: Tubagus Riko Rivanthio, S.Kom., M.Kom; 2017)

4. Diagram Urutan (*Sequence Diagram*)

Sequence Diagram menggambarkan perilaku pada sebuah skenario, diagram ini menunjukkan sejumlah contoh objek dan *message* (pesan) yang diletakkan diantara objek-objek ini di dalam *use case*, dimana sequence tersebut menggambarkan tampilan yang ditampilkan oleh program (Fitriana Harahap; 2017).

Sequence Diagram dapat digambarkan dengan simbol-simbol seperti pada tabel II.4.

Tabel II.4. Simbol Diagram Urutan

Gambar	Keterangan
	Object merupakan instance dari sebuah class dan dituliskan secara horizontal digambarkan sebagai sebuah class (kontak) dengan nama objeck di dalamnya yang diawali dengan sebuah titik koma.
	Actor juga dapat berkomunikasi dangan object, maka actor juga dapat diurutkan sebagai kolom simbol Actor sama dengan simbol pada *Use Case Diagram*.
	Lifeline mengindikasikan keberadaan sebuah object dalam batas waktu notasi untuk Lifeline adalah garis putus – putus vertical yang ditarik dari sebuah object
	Activation dinotasikan sebagai sebuah kotak segi empat yang digambarkan pada sebuah *Lifeline*. Activation mengindikasikan sebuah object yang akan melakukan sebuah aksi.
	Message digambarkan dengan anak panah horizontal antara Activation. Message mengindikasikan komunikasi antara object.

(Sumber: Agus Waluyo & Aang Munawar ; 2017)