BAB II
LANDASAN TEORI

2.1 Tinjauan Studi

Dalam penelitian ini akan digunakan tinjauan studi yang nantinya dapat mendukung penelitian, berikut ini merupakan tinjauan studi yang diambil:

1. (Andayani & Perwitasari, 2014), dengan judul "Penentuan rute terpendek pengambilan sampah di Kota Marauke menggunakan Algoritma Dijkstra". Penelitian ini dilakukan agar dapat meningkatkan kualitas pendistribusian sampah yang terkoordinasi dengan baik dan menimasi ongkos distribusi armada sehingga dapat menentukan total jarak terpendek armada yang ditempuhnya.
2. (Budihartanti & Pandiangan, 2016), dengan judul "Rancang bangun aplikasi android pencarian rumah sakit di Jakarta menggunakan algoritma dijkstra". Penelitian ini dilakukan oleh Penulis agar dapat menemukan lokasi rumah sakit yang ada di Jakarta ketika seseorang mengalami kecelakaan agar bisa lebih mudah untuk mengetahui lokasi rumah sakit tersebut.
3. (Retnani et al., 2015), dengan judul "Pencarian SPBU terdekat dan penentuan jarak terpendek menggunakan algoritma dijkstra (studi kasus di Kabupaten Jember" Penelitian ini membahas tentang mencari lokasi terdekat untuk mencari SPBU untuk menentukan jalur terpendek guna untuk memberikan efisiensi penggunaan bahan bakar pada kendaraan.
4. (Ismantohadi & Iryanto, 2018), dengan judul "Penerapan algoritma dijkstra untuk penentuan jalur terbaik evakuasi tsunami (studi kasus: Kelurahan Sanur Bali)" Penelitian ini bertujuan untuk mencari rute/jalur evakuasi terbaik di kelurahan tersebut. Pencarian jalur evakuasi terbaik tersebut dicari dengan

algoritma Dijkstra. Dalam penelitian ini, hasil jalur evakuasi terbaik dikelompokkan berdasarkan letak tempat evakuasi dan area aman.

5. (Ekasari, 2017), dengan judul "Penerapan algoritma dijkstra untuk menemukan lintasan terpendek pada pengiriman barang PT Kharisma Suma Jaya Sakti" pada penelitian ini peneliti menentukan lintas terpendek menggunakan Algoritma Dijkstra PT Kharisma Suma Jaya Sakti dapat meminimalisir biaya dan waktu yang diperlukan dalam pengiriman barang.
6. (Sulistiani & Wibowo, 2018), dengan judul "Perbandingan Alogoritma A* dan Dijkstra dalam Pencarian Kecamatan dan Kelurahan di Bandar Lampung" pada penelitian ini menentukan rute terpendek menggunakan algoritma A* dan dijkstra agar dapat membantu masyarakat di daerah Bandar Lampung untuk mengurus keperluan yang berhubungan dengan kantor Kecamatan dan Kelurahan.

Berdasarkan tinjauan studi literatur yang dipaparkan diatas maka penulis menjelaskan bahwa yang menjadi perbedaan pada penelitian sebelumnya dan penelitian yang akan diusulkan ialah perbedaan objek penelitian dan manfaat tambahan selain mencari lokasi juga memberikan informasi mengenai rute terpendek untuk mencari puskesmas terdekat yang ada di Kota Bandar Lampung ketika seseorang mengalami keadaan darurat dan akan berobat ke puskesmas agar lebih mudah untuk mengetahui lokasi puskesmas terdekat dan memberikan waktu yang efisien.

2.2 Algoritma Dijkstra

Algoritma Dijkstra dikembangkan oleh ilmuwan Belanda Edsger Dijkstra pada tahun 1959. Dalam jurnal (Miglani, 2016), *Dijkstra's Algorithm is a search algorithm that computes the single-source shortest path problem for a graph with nonnegative edge path costs, producing a shortest path tree*. Menurut kutipan diatas Algoritma Dijkstra adalah algoritma pencarian yang menghitung lintasan terpendek menggunakan grafik dengan bobot non-negatif, sehingga dapat menemukan jalur terpendek optimal yang merupakan perhitungan untuk masalah lintasan terpendek. Lintasan terpendek untuk sebuah *graph* terhubung berbobot. Dalam jurnal (Yao, Biyuan, 2016), Dijkstra *algorithm is well recognized as an efficient technique to address shortest path problem. It could be adopted to compute the distance among one node to the rest, thus enable to calculate the optimized results, which is a greedy algorithm to single-source shortest path problem.* Menurut kutipan diatas Algoritma Dijkstra ini juga diakui sebagai teknik yang efisien untuk mengatasi masalah lintasan terpendek. Sehingga dapat menghitung jarak antara satu node ke yang lain, serta memungkinkan untuk menghitung hasil optimal, yang merupakan perhitungan untuk masalah lintasan terpendek.

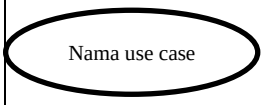
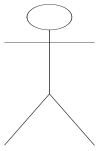
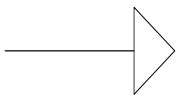
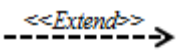
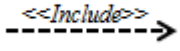
2.3 *Unified Modeling Language* (UML)

UML adalah diagram berbentuk grafik yang menunjukkan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Sebuah diagram merupakan bagian dari suatu *view* tertentu dan ketika digambarkan biasanya dialokasikan untuk *view* tertentu (A.S, Rosa, 2016). Adapun jenis diagram antara lain:

2.3.1 *Use Case Diagram*

Use case atau diagram use case merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeprisikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu (A.S, Rosa, 2016).

Tabel 2.1 Simbol-simbol *Use Case Diagram*

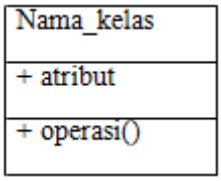
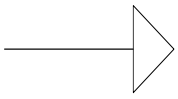
No.	Notasi	Keterangan	Simbol
1.	***Use case***	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saking bertukar pesan antar unit atau actor	
2.	**Aktor/ *Actor***	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi tang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang	
3.	**Asosiasi/ *Assocation***	Komunikasi antara factor dan use case yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor	
4	**Generalisasi/ *Generaliztion***	Hubungan generaliasai dan spesialisasi (umum – khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.	
5	**Ekstensi/ *Extend***	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tamahan itu.	
6	***Include***	*Use case* yang ditambahkan akan selalu dipanggil saat *use case* tambahan dijalankan.	

Sumber : (A.S, Rosa, 2016)

2.3.2 *Class Diagram*

Diagram kelas *atau Class diagram* menggambarkan struktur dari sistem dari segi pendefinisian kelas kelas yang akan dibuat untuk membangun sistem (A.S, Rosa, 2016).

Tabel 2.2 Simbol *Class* Diagram

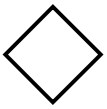
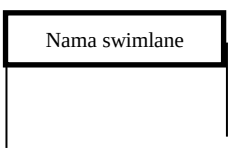
No	Notasi	Keterangan	Simbol
1.	***Kelas***	Kelas pada struktur sistem	
2.	**Antar muka/ *Interface***	Sama dengan konsep *interface* dalam pemograman berorientasi objek	
3.	**Asosiasi/ *Association***	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*	
4.	***Asosiasi berarah/ directed association***	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*	
5	***Generalisasi***	Relasi antar kelas dengan makna generalisasi spesialisasi (umum khusus)	
6	**Kebergantunga/ *dependency***	Relasi antar kelas dengan makna ketergantungan antar kelas	
7	**Agresi/ *aggregation***	Relasi antar kelas dengan makna semua – bagian (*whole-part*)	

Sumber : (A.S, Rosa, 2016)

2.3.3 *Activity Diagram*

Activity Diagram ini menggambarkan *workflow* (aliran kerja) atau aktifitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktifitas mengggambarkan aktifitas sistem bukan apa yang dilakukan aktor, jadi aktiftas yang dapat dilakukan oleh sistem (A.S, Rosa, 2016).

Tabel 2.3 Simbol *Activity Diagram*

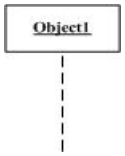
No.	Notasi	Keterangan	Simbol
1.	**Status awal**	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal	
2.	**Aktivitas**	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja	
3.	**Status akhir**	Status akhir yang dilakukan sistem, sebuah diagram aktifitas memiliki sebuah status akhir	
4.	**Percabangan / *Decision***	Asosiasi percabangan Dimana jika ada pilihan aktifitas lebih dari satu	
5.	**Penggabungan / *join***	Asosiasi penggabungan dimana lebih dari satu aktifitas digabungkan menjadi satu	
6.	***Swimlane***	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktifitas yang terjadi	

Sumber : (A.S, Rosa, 2016)

2.3.4 *Sequence Diagram*

Sequence Diagram digunakan untuk menggambarkan perilaku objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek pada sebuah *scenario* dan sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem (A.S, Rosa, 2016).

Tabel 2.4 Simbol *Squence* Diagram

No.	Notasi	Keterangan	Simbol
1.	*Object*	*Object* adalah *instance* dari sebuah class yang dituliskan tersusun secara horizontal diikuti *lifeline*	
2.	*Activation*	Indikasi dari sebuah objek yang melakukan suatu aksi	
3.	*Lifeline*	Indikasi keberadaan sebuah objek dalam basis waktu	
4.	*Message*	Indikasi untuk komunikasi antar *object*	
5.	*Self-Message*	Komunikasi kembali kedalam *object* itu sendiri	

Sumber : (A.S, Rosa, 2016)

2.4 *Global Positioning Sistem (GPS)*

Global Positioning Services (GPS) GPS adalah sebuah sistem navigasi satelit yang menyediakan informasi lokasi dan waktu dalam berbagai kondisi cuaca, dimanapun diatas permukaan bumi, sepanjang masih menerima sinyal GPS yang dipancarkan dari satelite (Marjuki, 2016). GPS saat ini telah dimanfaatkan dalam berbagai bidang tidak hanya dalam kegiatan survei dan pemetaan, tetapi juga kegiatan lain seperti kerekayasaan, olahraga, navigasi, transportasi, kebencanaan dan lain-lain. Menurut (Marjuki, 2016) aplikasi GPS dapat dikelompokkan menjadi lima yaitu :

1. Lokasi Penentuan posisi koordinat merupakan aplikasi paling dasar dari GPS. Contoh kegiatan penentuan lokasi dalam kehidupan sehari-hari misalnya, pemetaan titik dan penunjuk posisi.
2. Navigasi atau petunjuk arah dan jalur menuju tempat tertentu sekarang sudah lazim menggunakan GPS dibanding alat konvensional seperti kompas. Berbagai sarana dan kegiatan transportasi sudah menggunakan GPS.
3. *Tracking* adalah proses monitoring dan perekaman sepanjang perjalanan menuju lokasi tujuan.
4. *Mapping Survei* dan pemetaan merupakan salah satu pengguna utama GPS. Dengan menggunakan GPS, pemetaan dan ploting object dipermukaan bumi dapat langsung dilakukan tanpa harus mengintepretasi dan melihat posisi objek tersebut berdasarkan referensi tertentu.
5. *Timing* Penentuan waktu jeda sekarang juga menggunakan GPS sebagai salah satu sumber informasi atau referensi. Jam atom yang terpasang disatelit dapat

menjadi standar penunjuk waktu untuk berbagai kepentingan atau aplikasi yang memerlukan kepresisian waktu diseluruh dunia.

2.5 Metode Pengembangan Sistem

Menurut (Atin & Saputra, 2015), dalam jurnalnya menjelaskan bahwa, *Rational Unified Process* (RUP) merupakan suatu metode rekayasa perangkat lunak yang dikembangkan dengan mengumpulkan berbagai *best practises* yang terdapat dalam industri pengembangan perangkat lunak. Ciri utama metode ini adalah menggunakan *use-case driven* dan pendekatan iteratif untuk siklus pengembangan perangkat lunak. RUP menggunakan konsep *object oriented*, dengan aktifitas yang berfokus pada pengembangan model dengan Fase-fase yang dilakukan peneliti dalam metode RUP ini antara lain:

1) *Inception* (Permulaan)

   Pada tahap ini pengembang mendefinisikan batasan kegiatan, melakukan analisis kebutuhan *user*, dan melakukan perancangan awal perangkat lunak (perancangan arsitektural dan *use case*).

2) *Elaboration* (Perluasan/Perencanaan)

   Pada tahap ini dilakukan perancangan perangkat lunak mulai dari menspesifikasikan *fitur* perangkat lunak hingga perilisan prototipe.

3) *Construction* (Konstruksi)

   Pengimplementasian rancangan perangkat lunak yang telah dibuat dilakukan pada tahap ini. Pada akhir tahap ini, perangkat lunak versi akhir yang sudah disetujui administrator dirilis beserta dokumentasi perangkat lunak.

4) *Transition* (Transisi)

Instalasi, *deployment* dan sosialisasi perangkat lunak dilakukan pada tahap ini. Aktifitas pada tahap ini termasuk pada pemeliharaan dan pengujian.

2.6 Sistem Informasi Geografis

Menurut (Adil, 2017), Sistem Informasi Geografis, atau dalam Bahasa Inggris lebih dikenal dengan Geographic Information System adalah suatu sistem berbasis komputer yang digunakan untuk mengelolah dan menyimpan data atau informasi yang bereferesi geografis [2 SIG adalah sistem yang menggunakan informasi digital yang didapatkan dari metode pembuatan data digital. Metode pembuatan yang umum digunakan adalah *digitization*, yaitu peta cetak atau rencana survey yang ditransfer ke dalam bentuk media digital menggunakan program komputer (*Computer Aided Drafting, CAD*) serta kapabilitas *georeferencing*. Berdasarkan teknologi dan implementasinya sistem informasi geografis dapat dikategorikan dalam 3 (tiga) aplikasi, yaitu SIG berbasis desktop (desktop SIG), SIG berbasis web (web SIG) dan SIG berbasis mobile (*mobile* SIG). Ketiganya meskipun berbeda platform saling berhubungan satu dengan yang lainnya.

2.7 *Google Maps API*

Google Maps merupakan layanan dari google yang mempermudah penggunanya untuk melakukan kemampuan pemetaan untuk aplikasi yang dibuat. Sedangkan google Maps API memungkinkan pengembangan untuk mengintegrasikan Google Maps kedalam situs web. Dengan menggunakan *google Maps* API memungkinkan untuk menanamkan situs eksternal, dimana situs data tertentu dapat dilakukan *overlay* (Safaat, 2015).

2.8 Pengujian Sistem (ISO 9126)

Kualitas perangkat lunak dapat dinilai melalui ukuran-ukuran dan metode-metode tertentu, serta melalui pengujian-pengujian software. Salah satu tolak ukur kualitas perangkat lunak adalah ISO 9126, yang dibuat oleh *International Organization for Standardization (ISO)* dan *International Electrotechnical Commission (IEC)*. Standar ISO 9126 telah dikembangkan dalam usaha untuk mengidentifikasi atribut-atribut kunci kualitas untuk perangkat lunak komputer (Al-qutaish, 2010). Faktor kualitas menurut ISO 9126 meliputi enam karakteristik kualitas sebagai berikut:

1. *Functionality* (Fungsionalitas). Kemampuan perangkat lunak untuk menyediakan fungsi sesuai kebutuhan pengguna, ketika digunakan dalam kondisi tertentu.
2. *Reliability* (Kehandalan). Kemampuan perangkat lunak untuk mempertahankan tingkat kinerja tertentu, ketika digunakan dalam kondisi tertentu.
3. *Usability* (Kebergunaan). Kemampuan perangkat lunak untuk dipahami, dipelajari, digunakan, dan menarik bagi pengguna, ketika digunakan dalam kondisi tertentu.
4. *Efficiency* (Efisiensi). Kemampuan perangkat lunak untuk memberikan kinerja yang sesuai dan relatif terhadap jumlah sumber daya yang digunakan pada saat keadaan tersebut.
5. *Maintainability* (Pemeliharaan). Kemampuan perangkat lunak untuk dimodifikasi. Modifikasi meliputi koreksi, perbaikan atau adaptasi terhadap perubahan lingkungan, persyaratan, dan spesifikasi fungsional.

6. *Portability* (Portabilitas). Kemampuan perangkat lunak untuk ditransfer dari satu lingkungan ke lingkungan lain.

Masing-masing karakteristik kualitas perangkat lunak model ISO 9126 dibagi menjadi beberapa sub-karakteristik kualitas, yaitu:

Tabel 2.5 Karakteristik Kualitas Perangkat Lunak Model ISO 9126

Karakteristik	**Sub- karakteristik**	**Deskripsi**
Functionality	*Suitability*	Kemampuan perangkat lunak untuk menyediakan serangkaian fungsi yang sesuai untuk tugas-tugas tertentu dan tujuan pengguna.
	Accuracy	Kemampuan perangkat lunak dalam memberikan hasil yang presisi dan benar sesuai dengan kebutuhan.
	Security	Kemampuan perangkat lunak untuk mencegah akses yang tidak diinginkan, menghadapi penyusup (*hacker*) maupun otorisasi dalam modifikasi data.
	Interoperability	Kemampuan perangkat lunak untuk berinteraksi dengan satu atau lebih sistem tertentu.
	Compliance	Kemampuan perangkat lunak dalam memenuhi standar dan kebutuhan sesuai peraturan yang berlaku.
Reliability	*Maturity*	Kemampuan perangkat lunak untuk menghindari kegagalan sebagai akibat dari kesalahan dalam perangkat lunak.

Karakteristik	Sub- karakteristik	Deskripsi
	Fault tolerance	Kemampuan perangkat lunak untuk mempertahankan kinerjanya jika terjadi kesalahan perangkat lunak
	Recoverability	Kemampuan perangkat lunak untuk membangun kembali tingkat kinerja ketika terjadi kegagalan sistem, termasuk data dan koneksi jaringan.
Usability	*Understandibility*	Kemampuan perangkat lunak dalam kemudahan untuk dipahami.
	Learnability	Kemampuan perangkat lunak dalam kemudahan untuk dipelajari.

Tabel 2.5 Karakteristik Kualitas Perangkat Lunak Model ISO 9126 (Lanjutan)

Karakteristik	Sub- karakteristik	Deskripsi
	Operability	Kemampuan perangkat lunak dalam kemudahan untuk dioperasikan.
	Attractiveness	Kemampuan perangkat lunak dalam menarik pengguna.
Efficiency	*Time behavior*	Kemampuan perangkat lunak dalam memberikan respon dan waktu pengolahan yang sesuai saat melakukan fungsinya.
	Resource Behavior	Kemampuan perangkat lunak dalam menggunakan sumber daya yang dimilikinya ketika melakukan fungsi yang ditentukan.
Maintainability	*Analyzability*	Kemampuan perangkat lunak dalam mendiagnosis kekurangan atau penyebab kegagalan.
	Changeability	Kemampuan perangkat lunak untuk dimodifikasi tertentu.
	Stability	Kemampuan perangkat lunak untuk meminimalkan efek tak terduga dari modifikasi perangkat lunak.

Karakteristik	Sub- karakteristik	Deskripsi
	Testability	Kemampuan perangkat lunak untuk dimodifikasi dan divalidasi perangkat lunak lain.
Portability	*Adaptability*	Kemampuan perangkat lunak untuk diadaptasikan pada lingkungan yang berbeda-beda.
	Instalability	Kemampuan perangkat lunak untuk diinstal dalam lingkungan yang berbeda-beda.
	Coexistence	Kemampuan perangkat lunak untuk berdampingan dengan perangkat lunak lainnya dalam satu lingkungan dengan
	Replaceability	Kemampuan perangkat lunak untuk digunakan sebagai sebagai pengganti perangkat lunak lainnya.

Sumber : (Al-qutaish, 2010)