BAB II

TINJAUAN PUSTAKA

II.1 Penelitian Sebelumnya

Dalam penelitian sebelumnya, yang dilakukan oleh Martin Nugroho Parapat, Deddy Subianto, Cahya Rahmad dengan judul "Rancang Bangun Aplikasi Pencarian Rute Terpendek Jasa Kiriman Barang Berbasis Mobile Dengan Metode Algoritm Adijkstra" menghasilkan aplikasi yang dapat membantu pegawai kurir PT.Pos Indonesia dalam menentukan jarak terpendek dan pendapatan tiap wilayah dengan rute yang telah di tentukan.

Penelitian selanjutnya yaitu dari Milwansyah Siregar tahun 2018 dengan judul "Implementasi Algoritma Dijkstra Untuk Pencarian Rute Terpendek Kantor Polisi Di Wilayah Kota Medan Berbasis Mobile". Menghasilkan sistem in*Form* asi lokasi kantor polisi di kota Medan berbasis mobile. Sehingga seseorang yang ingin mengunjungi kantor polisi di kota medan dapat dengan cepat sampai ke lokasi karena aplikasi yang dibuat oleh Milwansyah Siregar tersebut.

Penelitian selanjutnya adalah penelitian dari sutriono dengan judul "Implementasi Algoritma Dijkstra Pada Aplikasi Pencarian Bengkel Tambal Ban Terdekat Di Kota Medan Berbasis Android". Menghasilkan aplikasi yang mempermudah seseorang yang tiba-tiba terkena musibah ban bocor dalam menemukan lokasi bengkel terdekat tanpa harus bertanya kepada warga sekitar yang akan memperlambat waktu dalam menemukan bengkel dan penanganannya.

Penelitian selanjutnya dari Utti Marina Rifanti degan judul "Pemilihan Rute Terbaik Menggunakan Algoritma Dijkstra Untuk Mengurangi Kemacetan Lalu

Lintas Di Purwokerto" Menghasilkan aplikasi yang dapat menentukan jalur alternatif yang dapat dilalui pengendara untuk menghindari terjadinya kemacetan di ruas jalan tertentu.

Yang kelima penelitian dari Ika Ismantohadi Iryanto dengan judul "Penerapan Algoritma Dijkstra Untuk Menentukan Jalur Terbaik Evakuasi Tsunami Studi Kasus Kelurahan Santu Bali". Dengan hasil dari penelitian yaitu jalur evakuasi terbaik dikelompokkan berdasarkan letak tempat evakuasi dan area aman. Sehingga dapat meminimalisir banyaknya korban.

Yang terakhir penelitian dari Faisal, Usman Syamsudin dengan judul "Aplikasi Jasa Pemesanan Digital Printing Berbasis Web". Hasil dari penelitian ini yaitu untuk mempermudah proses pemesanan produk digital printing. Segingga masyarakat tidak perlu repot-repot mendatangi gerai karena bisa dipesan secara *online* juga dapat menghemat waktu dan tenaga.

Kemudian hasil dari penelitian penulis dengan judul "Ranacang Bangun Aplikasi Pemesanan Ambulan Secara Online Untuk Situasi Darurat Dengan Metode Algoritma Dijkstra Berbasis Android" yaitu pengguna dapat memesan ambulan dengan mudah dan cepat hanya perlu tekan tombol pesan maka akan langsung menelepon pihak Rumah Sakit dan ambulan akan segera datang. Namun jika pengguna hanya memerlukan in*Form* asi tentang Rumah Sakit yang paling dekat dengan lokasinya bisa melihatnya melalui aplikasi tersebut dan akan muncul navigasinya. Dari aplikasi yang penulis buat memiliki beberapa keunggulan antara lain, pengerjaannya menggunakan algoritma dijkstra sehingga jalur untuk *maps* terlihat jelas, menggunakan database SqlLite sehingga pengguna dapat menggunakan saat offline, satu aplikasi hanya dapat

digunakan oleh satu *user* sehinnga keamanan terjaga dari pengguna yang iseng, bisa memesan Ambulan Rumah Sakit dengan cepat saat keadaan darurat, dan aplikasi yang mudah digunakan dan dipahami.

II.2 Landasan Teori

Adapun beberapa hal yang penting dalam penelitian ini antara lain sebagai berikut :

II.2.1 Rumah Sakit

Menurut S. Iskandar (2016) Rumah Sakit sebagai salah satu sarana kesehatan yang memberikan pelayanan kesehatan kepada masyarakat memiliki peran yang sangat strategis dalam mempercepat peningkatan derajat kesehatan masyarakat, oleh karena itu Rumah sakit memberikan pelayanan yang bermutu yang memuaskan bagi pasiennya sesuai standar yang telah ditetapkan dan dapat menjangkau seluruh lapisan masyarakatnya. Menurut UU Kesehatan, pelaksanaan pelayanan kesehatan harus mendahulukan pertolongan keselamatan nyawa pasien dibanding kepentingan lainnya. Konsekuensi logis dari hal tersebut adalah bahwa penyelenggaraan upaya kesehatan lebih banyak berorientasi pada aspek sosial kemanusiaan sebagai sarana untuk pengabdian terhadap kepentingan masyarakat. Kepentingan masyarakat dalam hal ini adalah untuk mendapatkan pelayanan kesehatan yang berkualitas dan terjangkau.

II.2.2. Ambulan

Menurut M.F. Jauhari, M. Arsyad, R. S. Maryati, and P. N. Banjarmasin (2018) Penggunaan Ambulan yaitu sebagai unit transportasi layanan sosial dalam hal transportasi gawat darurat medis atau keperluan lainnya cukup memiliki peran penting di masyarakat. Ambulan adalah kendaraan transportasi gawat darurat medis khusus orang sakit atau cedera yang digunakan untuk membawanya dari satu tempat ke tempat lain guna perawatan lebih lanjut. Istilah ambulan digunakan untuk menerangkan kendaraan yang digunakan untuk membawa peralatan medis kepada pasien di luar rumah sakit atau memindahkan pasien ke rumah sakit untuk perawatan lebih lanjut. Kendaraan ini dilengkapi dengan sirene dan lampu berwarna merah dan biru gawat darurat agar dapat menembus kemacetan lalu lintas.

II.2.3. Pencarian 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.

Menurut J. V. Ginting dan E. S. Barus (2018) Waktu dapat dikaitkan dengan jarak tempuh, semakin pendek jarak tempuh maka semakin singkat waktu yang dibutuhkan untuk menempuh jarak tersebut. Penghitungan rute terpendek memegang peranan penting dalam kehidupan sehari-hari karena harus dilakukan dalam waktu singkat dan pada saat itu juga agar segera dapat diketahui rute mana yang paling pendek untuk dilewati. Dengan melewati rute terpendek dapat membuat mobilitas sehari-hari menjadi lebih efisien. Untuk mencapai suatu tempat yang dituju terkadang seseorang tidak mengetahui terdapat jalur terdekat untuk

mencapai tujuannya. Jika seseorang mengetahui jalur terdekat tersebut akan dapat mempersingkat waktu tempuh yang di milikinya.

II.2.4.*Algoritme Dijkstra*

Menurut Marsudi (2016) *Algoritma Dijkstra* adalah sebuah algoritma untuk menentukan lintasan terpendek dalam graf tak berarah(atau graf berarah) berbobot tanpa mengenumerasi secara tepat semua lintasan yang mungkin. algoritma ini didasarkan pada sebuah metode yang dikenal sebagai pemrograman dinamik. algoritma dijkstra menentukan lintasan terpendek diantara pasangan-pasangan titik dalam suatu *graf.*

Menurut Hary Cahyono (2019) Sesuai dengan nama penemunya (*Edsger Dijkstra*), algoritma *Dijkstra* digunakan untuk memecahkan permasalahan jarak terpendek (shortest path proble) untuk sebuah graph berarah. Menurut info dari anikel program MT' Binus University, "Algoritma ini dipublikasikan pada tahun 1959 di jurnal *Numerische Mathematik* yang berjudul *'A Nore on Two Problems in Conne_rion with Graphs*' dan dianggap sebagai algoritrna *greedy*".

"Permasalahan rute terpendek dari sebuah titik ke akhir titik lain adalah sebuah masalah klasik optimasi yang banyak digunakun untuk menguji sebuah algoritmu yang diusulkan. Permasalohan rule terpendek dianggap cukup baik untuk mewakili masalah optimisasi. karena permasalahannya mudah dimengerti (hanya menjumlahkan seluruh *edge* yang dilalui namun memiliki banyak solusi”

Algoritma yang ditemukan djikstra untuk mencari jalur terpendek merupakan algoritma yang lebih efisien dibandingkan algoritma Warshall, meskipun implementasinya juga lebih sukar (sulit) diselesaikan. Misalkan G adalah graf berarah yang berlabel dengan titik-titik V(G)={v1,v2,...,vn} dan jalur

terpendek yang dicari adalah dari v1 ke vn. Algoritma Djikstra dimulai dari titik v1. Dalam literasinya, algoritma akan mencari satu titik yang jumlah bobotnya dari titik 1 terkecil. Titik-titik yang terpilih dipisahkan dan titik-titik tersebut tidak diperhatikan lagi dalam literasi berikutnya.

Misalkan:

V(G)={v1,v2,.....,vn}L	= Himpunan titik-titik V(G) yang sudah terpilih dalam jalur terpendek
D(j)	= Jumlah bobot terkecil dari v1 ke vj
w(i,j)	= Bobot garis dari titik v1 ke vj
w*(1,j)	= Jumlah bobot terkecil dari v1 ke vj

Menurut algortima diatas, jalur terpendek dari titik v1 ke vn adalah melalui titik-titik dalam L secara berurutan, dan jumlah bobot terkecilnya adalah D(n). Untuk menghitung *weight* (bobot) nilai minimum menggunakan persamaan berikut.

Weight(w) = min(Dastvalue,Markvalue + EdgeValue)

Dimana DastValue = Nilai dalam node tujuan

MarkValue = Nilai dalam node awal

EdgeValue = Nilai dari sisi yang menghubungkan node

Kompleksitas Waktu Algoritma Dijktra Untuk himpunan simpul serta sisi dengan jumlah simpul sebesar V dan jumlah sisi sebesar E, dapat ditentukan kompleksitas waktu Algoritma Dijkstra dengan notasi Big-O, yaitu O (| V | 2 + | E |) = O (| V | 2), jika Algoritma menyimpan simpul dan sisi dalam bentuk list berkait ataupun array. Algoritma dapat lebih efisien dengan menyimpan graf dalam bentuk daftar dan menggunakan *binary heap* atau *fibonacci heap* sebagai *priority queue*. Didapatkan dalam notasi Big-O, yaitu O ((| E | + | V |) log | V |).

II.2.5. *Graf*

Graf adalah objek dasar pelajaran dalam teori graf. Dalam bahasa sehari-hari, sebuah graf adalah himpunan dari objek-objek yang dinamakan *titik*, *simpul*, atau *sudut* dihubungkan oleh penghubung yang dinamakan *garis* atau *sisi*. Dalam graf yang memenuhi syarat, di mana biasanya *tidak berarah*, sebuah garis dari titik *A* ke titik *B* dianggap sama dengan garis dari titik *B* ke titik *A*. Dalam *graf berarah*, garis tersebut memiliki arah. Pada dasarnya, sebuah graf digambarkan dengan bentuk diagram sebagai himpunan dari titik-titik (sudut atau simpul) yang digabungkan dengan kurva (garis atau sisi).

II.3. *Android*

Menurut Hartati et al (2017) android adalah sebuah sistem operasi pada handphone yang bersifat terbuka dan berbasis Linux. *Android* menyediakan plat*Form* terbuka (*open source*) sehingga memudahkan bagi para pengembang untuk menciptakan aplikasi mereka sendiri, android awalnya dikembangkan oleh *Android, Inc*, dengan dukungan Google, yang kemudian android dibeli oleh Google pada tahun 2005.

II.3.1. *Android Software Development Kit (SDK)*

Menurut Jubilee Enterprise (2015) *Android Software Development Kit (SDK*) adalah tool Android yang terintegrasi dengan android studio yang digunakan untuk mengontrol instalasi SDK. dengan tool ini anda dapat melakukan proses update dan install komponen-komponen yang diperlukan dalam Android Studio.

II.3.2. *Library* dan *Android runtime*

Menurut Yudho Yudhanto dan Ardhi Wijayanto (2019) Setiap aplikasi berjalan dalam prosesnya sendiri dan dengan *instance Android runtime* sendiri, yang memungkinkan beberapa mesin sekaligus *virtual* pada perangkat bermemori rendah. Android juga menyertakan rangkaian waktu proses inti yang menyediakan sebagian besar fungsionalitas bahasa pemrograman java, termasuk beberapa fitur bahasa java 8 yang digunakan *framework java API*. Banyak layanan dan komponen sistem android inti dibangun dari kode asli yang memerlukan pustaka asli yang ditulis dalam C dan C++. Pustaka asli tersebut tersedia untuk aplikasi melalui kerangka kerja Java API.

II.3.3. GPS

Menurut Yudho Yudhanto dan Ardhi Wijayanto (2019) Sistem operasi Android mendukung GPS yang memungkinkan developer untuk mengakses lokasi pengguna. Contoh aplikasi yang memanfaatkan GPS adalah aplikasi peta (*maps*) yang menunjukkan lokasi pengguna dan memberikan petunjuk untuk menuju suatu lokasi.

II.4. Metode *Waterfall*

Pada pembuatan program ada tahapan prosedur yang harus dilakukan yaitu dengan melakukan SLDC (*System Development Life Cycle*). Menurut Rosa A. S dan M. Shalahuddin (2018) SLDC atau *Software Development Life Cycle* atau sering disebut juga *System Development Life Cycle* adalah proses mengembangkan atau mengubah sistem perangkat lunak dengan menggunakan model-model dan metodologi yang digunakan orang untuk mengembangkan sistem-sistem perangkat

lunak sebelumnya(berdasarkan *best practice* atau cara-cara yang sudah teruji baik). Seperti halnya proses metamorfosis pada kupu-kupu, untuk menjadi kupu-kupu yang indah maka dibutuhkan beberapa tahap untuk dilalui, sama halnya dengan membuat perangkat lunak, memiliki daur tahapan yang dilalui agar menghasilkan perangkat lunak yang berkualitas.

SLDC memiliki banyak model antara lain model *waterfall*, Model SLDC air terjun (*waterfall)* sering juga disebut model sekuensial linier(*sequensial linier*) atau alur hidup klasik(*clasic life cycle*). Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengodean, pengujian, dan tahap pendukung (*support*) (Rosa A. S dan M. Shalahuddin. 2018).

II.5. UML

UML (*unified Modeling Language)* adalah salah standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement,* membuat analisis & desain, serta menggambarkan arsitektur dalam pemrograman dan berorientasi objek.

Berikut ini penjelasan singkat dari pembagian kategori diagram UML :

- *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
- *Behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.

- *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem (Rosa A. S dan M. Shalahuddin. 2018).

Dalam penelitian ini peneliti akan menggunakan *Class Diagram, Activity Diagram,Use Case Diagram* dan *Sequence Diagram.* Berikut ini penjelasannya, Menurut Rosa A. S dan M. Shalahuddin. (2018).

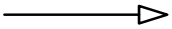
1. *Use Case Diagram*

   Use Case atau diagram *use case* merupakan pemodelan untuk kelakuan (behavior) sistem in*Form* asi yang akan dibuat. *Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem in*Form* asi yang akan dibuat.

Tabel II.1 *Use Case Diagram*

No	Simbol	Nama	Keterangan
1		Aktor / *Actor*	Orang, proses, atau sistem lain yang berineraksi dengan sistem in*Form* asi yang akan dibuat di luar sistem in*Form* asi yang akan di buat sistem itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan

			orang : biasanya dinyatakan menggunakan kata benda diawal frase nama aktor.
2		Asosiasi / *Association*	Komunikasi antara aktor dan *use case* atau *use case* memiliki interaksi dengan aktor.
3	<<extend>>	Ekstensi / *extend*	Relasi *use case tambahan* ke sebuah *use case* dimana *use case* yang di tambahkan dapat berdiri sendiri walau tanpa *use case* tambahan.
4	Login	*Use Case*	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja diawal *frase* nama *use case.*

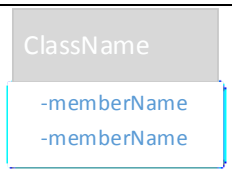
5		Generalisasi / *Generalization*	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
6		Menggunakan / *include / uses*	Relasi *use case* tambahan ke sebuah *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini.

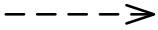
(*Sumber : Rekayasa Perangkat Lunak, 155-158*)

2. *Class Diagram*

Diagram kelas atau *Class Diagram* menggambarkan struktur sistem dari segi pendefinian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut *atribut* dan metode atau operasi.

Tabel II.2 *Class Diagram*

No	Simbol	Nama	Keterangan
1		Kelas	Kelas pada struktur sistem.
2		Antar muka / *Interface*	Sama dengan konsep *Interface* dalam pemrograman berorientasi objek
3		Asosiasi / *Association*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
4		Asosiasi Berarah / *DirectedAssociation*	Relasi antar kelas dengan makna kelas yang satu di gunakan oleh kelas yang lain,

			asosiasi biasanya juga disertai dengan *multiplicity.*
5		Generalisasi	Relasi antarkelas dengan makna generalisasi-spesialisasi (umum-khusus).
6		Kebergantungan / *Dependency*	Relasi antarkelas dengan makna kebergantungan antarkelas.
7		Agregasi / *Aggregation*	Relasi antarkelas dengan makna semua – bagian (*whole – part).*

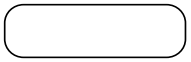
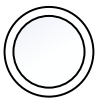
(*Sumber : Rekayasa Perangkat Lunak, 146-147)*

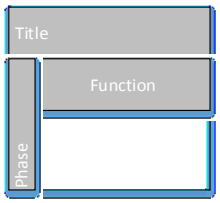
3. *Activity Diagram*

Diagram aktivitas atau *Activity Diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa

diagram aktivitas menggambatkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Tabel II.3 *Activity Diagram*

No	Simbol	Nama	Keterangan
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		Percabangan / *Decision*	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
4		Penggabungan / *join*	Asosiasi penggabungan dimana lebih dari datu aktivitas digabungkan menjadi satu.
5		Status Akhir	Status akhir yang dilakukan sistem, sebuah diagram aktivitas

			memiliki sebuah status akhir.
6		*Swimline*	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

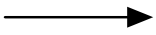
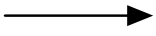
(*Sumber : Rekayasa Perangkat Lunak, 162-163*)

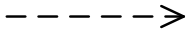
4. *Sequence Diagram*

Sequence Diagram merupakan permodelan untuk menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek, proses penggambaran diagram sekuen harus diketahui objek yang terlihat pada *use case* beserta metode yang dimiliki, sehingga menjadi objek.

Tabel II.4 *Sequence Diagram*

No	Simbol	Nama	Keterangan
1		Aktor / *Actor*	Orang, proses, atau sistem lain yang berineraksi dengan sistem in*Form* asi yang akan dibuat di luar sistem in*Form* asi yang akan di buat sistem itu sendiri,

			jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang : biasanya dinyatakan menggunakan kata benda diawal frase nama aktor.
2	Nama objek : nama kelas	Objek	Menyatakan objek yang berinteraksi pesan.
3		Waktu Aktif	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya.
4		Pesan Tipe *Create*	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat.
5	1 : nama metode() 	Pesan Tipe *Call*	Menyatakan suatu objek memanggil operasi/metode yang ada

			pada objek lain atau dirinya sendiri.
6	1 : masukan 	Pesan Tipe *Send*	Menyatakan bahwa suatu objek mengirimkan data/masukan/in*Form* asi ke objek lainnya, arah panah mengarah pada objek yang dikirimi.
7	1 : keluaran 	Pesan tipe *Return*	Menyatakan bahwa suatu objek yang telah menjalakan suatu operasi atau metode menghasilkan sesuatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.

(*Sumber : Rekayasa Perangkat Lunak, 165-167*)

II.7. *Database*

Database merupakan kumpulan in*Form* asi ini disimpan dengan cara yang sangat terstruktur. Dengan mengeksploitasi struktur tersebut, Anda mengakses dan

memodifikasi in*Form* asi dengan cepat dan tepat. Database sekarang ada dimana saja. Dan setiap database memerlukan suatu cara agar *user* bisa berinteraksi dengan in*Form* asi yang ada di dalamnya. Interaksi semacam itu dilakukan oleh DBMS (*Database Management System*) (Vivian Siahaan, Rismon Hasiholan Sianipar, 2019).

II.7.1. *SQLite*

SQLite merupakan mesin database yang mudah digunakan. Secara mendasar, *SQLite* merupakan merupakan database ringan yang dikhususksn untuk aplikasi-aplikasi berukuran kecil yang yang dapa disimpan pada satu *file disk*. Mesin database ini sangat populer yang digunakan pada telepon seluler, tablet, peralatan dan instrumentasi elektronis. *SQLite* tidak memerlukan proses *server* terpisah, dan tidak memerlukan konfigurasi apapun. (Vivian Siahaan, Rismon Hasiholan Sianipar, 2019).