BAB 2

LANDASAN TEORI

2.1 *Central Kitchen* (CK)

Central Kitchen merupakan nama lain dari Dapur merupakan suatu ruangan atau tempat khusus yang memiliki perlengkapan dan peralatan untuk mengelolah makanan dan untuk memasak makanan. Proses yang secara strategis mengelola pengadaan, pergerakan, dan penyimpanan material, suku cadang dan barang jadi beserta aliran informasi terkait melalui organisasi dan kanal kanal pemasarannya, dengan cara dimana keuntungan perusahaan, baik untuk saat ini maupun diwaktu yang akan datang, sehingga dapat dimaksimalkan dengan cara pemenuhan pesanan yang berbiaya efektif (*Christopher*, 2013). Logistik melibatkan proses perencanaan, implementasi, dan pengendalian, agar didapat suatu efisiensi akan biaya dan keefektifan proses penyimpanan bahan mentah, setengah jadi, barang jadi, dan informasi-informasi yang berhubungan, dari asal ke titik konsumsi dengan tujuan memenuhi kebutuhan konsumen (*The Council Logistics Management*).

Logistik merupakan fungsi yang melibatkan perpindahan, mengatur perpindahan barang, dan penyimpanan material dalam perjalanannya dari pengirim awal, melalui rantai pasok dan sampai ke pelanggan akhir. (*Donald Walters*, 2003. p. 3-4).

2.2 Sistem Informasi

Sistem merupakan kumpulan elemen-elemen yang saling terkait dan bekerja sama untuk memproses masukan (*input*) yang ditujukan kepada sistem tersebut dan

mengolah masukan tersebut sampai menghasilkan keluaran (*output*) yang diinginkan. Suatu sistem pada dasarnya sekelompok unsur-unsur yang erat hubungannya satu dengan yang lain yang berfungsi untuk mencapai tujuan tertentu. Sistem informasi memiliki makna sistem yang bertujuan menampilkan informasi. (Agustina, 2018)

2.3 Android

Android adalah sistem operasi berbasis Linux yang dirancang untuk perangkat bergerak layar sentuh seperti telepon pintar dan komputer tablet. Android awalnya dikembangkan oleh Android, Inc dengan dukungan finansial Google, yang kemudian membelinya pada tahun 2005. (*George,* 2018)

Android merupakan terobosan baru dalam bidang teknologi saat kini, dengan kemudahan pemakaiannya dan bersifat *open source* membuat peminat dari gadget ini semakin banyak dan sudah menjadi hal yang umum. Hampir semua vendor saat ini mengembangkan produknya dengan sistem operasi Android. Mulai dari pengembangan aplikasi yang dapat diunduh dengan mudah, hingga pengembangan sistem yang beragam. Selain itu, banyaknya aplikasi-aplikasi yang memudahkan para pengguna gadget smartphone untuk berkomunikasi dan menunjang kegiatan sehari-hari telah banyak ditawarkan di *Play Store*, aplikasi yang menjadi pusat dari segala aplikasi Android yang dapat dipasang pada smartphone Android dengan masing-masing kelebihan dari tiap aplikasi tersebut membuat para pengguna gadget smartphone banyak menggunakannya.

2.4 Android Studio

Android Studio adalah sebuah IDE untuk Android Development yang diperkenalkan google pada acara Google I/O 2013. Android Studio merupakan pengembangkan dari Eclipse IDE, dan dibuat berdasarkan IDE Java populer, yaitu IntelliJ IDEA. Android Stud io merupakan IDE resmi untuk pengembangan aplikasi Android. (*George*, 2018)

Inti dari Android Studio adalah editor kode cerdas, mampu mengcode completion dengan cerdas, refactoring, dan analisis code. Editor yang baik dalam membantu pengembang aplikasi Android agar lebih produktif.

Menurut pengembang Android Studio, *software* ini dilengkapi dengan *intellegent code editor* yang mampu mengolah dan menganalisis kode secara lengkap yang menjadikan *developer* semakin produktif. Selain itu pula, Android Studio dilengkapi dengan *Code Templates* dan Github integration yang memudahkan developer Android dalam mengembangkan aplikasi mereka dari sampel-sampel kode yang disediakan ataupun mengimpornya dari Github.

2.5 Jarak Terpendek (*Shortest Path Problem*)

Proses penghitungan rute terpendek merupakan proses dimana untuk mendapatkan jarak yang terpendek atau dengan biaya yang sangat kecil pada suatu rute dari node awal ke node tujuan dalam sebuah jaringan. Pada proses penghitungan rute terpendek terdapat dua macam proses yaitu proses pemberian label dan proses pemeriksaan node. Metode pemberian label adalah metode untuk memberikan identifikasi pada setiap node dalam jaringan.

Pada sebagian besar algoritma penghitungan rute terpendek, terdapat 3 label informasi yang dikelola untuk setiap node i pada proses pemberian label yaitu: label jarak d(i), parent node p(I), dan status node S(i). Algoritma untuk mencari jarak terpendek sudah banyak yang meneliti. Beberapa algoritma yang dapat digunakan untuk menyelesaikan penentuan jarak terpendek adalah algoritma dijkstra, algoritma bellmanford, algoritma a* , dan algoritma floyd–warshall.

2.6 Algoritma Greedy

Algoritma dijkstra merupakan salah satu bentuk algoritma greedy. Algoritma ini termasuk algoritma pencarian untuk menyelesaikan masalah jarak terpendek dengan satu tujuan pada sebuah lintasan yang tidak memiliki *cost* sisi *negative*. Algoritma dijkstra menggunakan *adjacent list* untuk merepresentasikan sebuah jalur yang akan dilewati.

Algoritma Dijkstra mencari lintasan terpendek dalam sejumlah langkah dengan menggunakan strategi greedy. Strategi greedy pada Algoritma Dijkstra Pada setiap langkah, ambil sisi yang berbobot minimum yang menghubungkan sebuah simpul yang sudah terpilih dengan sebuah simpul lain yang belum terpilih. Lintasan dari simpul asal ke simpul yang baru haruslah merupakan lintasan yang terpendek diantara semua lintasannya ke simpul-simpul yang belum terpilih.

Misalkan sebuah graf berbobot dengan n buah simpul dinyatakan dengan matriks ketetanggan M dengan simpul awal adalah a, dan dengan jarak dari simpul i diartikan sebagai jarak antara simpul a dan i, maka algoritma Djikstra akan menginitialisasikan nilai jarak awal dan memperbaikinya tahap demi tahap.

1. Inisiasikan suatu nilai jarak untuk setiap simpul dengan nilai nol untuk simpul awal dan nilai tak hingga untuk setiap simpul sisanya.

2. Tandailah seluruh simpul sebagai belum dikunjungi dan tentukan simpul a sebagai simpul saat ini.

3. Untuk simpul saat ini, perhitungkan seluruh tetangga langsungnya yang belum dikunjungi dan kalkulasikan jarak tentative dari simpul a. Jika jarak yang didapatkan lebih kecil daripada jarak yang sudah dicatat sebelumnya, maka jarak yang minimum akan disimpan.

4. Jika kita sudah selesai dengan pengecekan terhadap semua tetangga terdekat dari simpul saat ini, simpul ditandai sebagai sudah dikunjungi.

5. Sebuah simpul yang ditandai sebagai sudah dikunjungi tidak akan pernah diperiksa ulang dan jarak yang tercatat adalah akhir dan minimal.

6. Jika seluruh simpul sudah selesai dikunjungi, maka selesai; selain daripada itu pilih simpul dengan jarak minimum dari simpul a untuk ditetapkan sebagai simpul saat ini dan ulangi langkah 3.

2.7 Algoritma Bellman-ford

Algoritma bellman-ford dikembangkan oleh *Richard Bellman and Lester Ford, Jr.* Algoritma ini sangat mirip dengan algoritma dijkstra namun algoritma ini mampu menangani bobot negatif pada pencarian jarak terpendek pada sebuah graph berbobot. Algoritma bellman-ford merupakan pengembangan dari algoritma dijkstra, algoritma bellman-ford akan benar jika dan hanya jika graph tidak terdapat cycle dengan nilai bobot negatif yang dicapai dari sumber.

Ciri-ciri algoritma bellman-ford :

1. Mengeksekusi walaupun terdapat edge dengan bobot negatif.

2. Harus *directed edge* (jika tidak graph akan memilik cycle dengan bobot negatif).

3. Iterasi i menemukan seluruh shortest path dengan menggunakan i edge.

4. Dapat mendeteksi cycle dengan nilai bobot negatif jika ada.

Struktur dasar dari Bellman-Ford sangatlah menyerupai yang ada pada algoritma Dijkstra, akan tetapi dibandingkan dengan memilih simpul dengan bobot minimum yang belum digunakan secara greedy, algoritma Bellman-Ford secara sederhana melakukan pengecekan terhadap semua sisi dan melakukannya sejumlah |V| - 1 kali, dimana |V| adalah banyaknya simpul yang terdapat di dalam graf. Pengulangan ini memungkinkan jarak terpendek untuk dihitung secara akurat bertahap di dalam graf dengan kemungkinan suatu simpul dikunjungi dalam lintasan tersebut adalah sebanyak-banyaknya satu kali saja. Adapun kompleksitas waktu untuk keberjalanan algoritma Bellman-Ford dimana |V| dan |E| adalah banyaknya simpul dan sisi berturutan.

2.8 Java

Java menurut defenisi dari Sun adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer personal ataupun pada lingkungan jaringan. Java2 adalah generasi kedua dari java platform (generasi awalnya adalah Java Development Kit). (George, 2018)

2.9 PHP

PHP (*Hypertext Preprocessor*), merupakan bahasa pemrograman pada sisi server yang memperbolehkan programmer menyisipkan perintah – perintah perangkat lunak web server (Apache, IIS, atau apapun) akan dieksekusi sebelum perintah itu dikirim oleh halaman ke browser yang me-request-nya (Deddy Kusbianto, 2018).

2.10 Firebase

Firebase adalah API yang disediakan google untuk penyimpanan dan penyelarasan data ke dalam aplikasi Android, iOS, atau web. *Realtimedatabase* adalah salah satu fasilitas yang menyimpan data ke database dan mengambil data darinya dengan sangat cepat tetapi firebase bukan hanya *realtime database*, jauh lebih dari itu. Firebase memiliki banyak fitur seperti *authentication, database, storage, hosting,* pemberitahuan dan lain-lain. (George, 2018)

2.11 XAMPP

XAMPP adalah sebuah software yang berfungsi untuk menjalankan website berbasis PHP dan menggunakan pengolah data MySQL di komputer lokal. XAMPP berperan sebagai server web pada komputer. XAMPP juga dapat disebut sebuah CPanel server virtual, yang dapat membantu melakukan *preview* sehingga dapat memodifikasi website tanpa harus online atau terakses dengan internet. XAMPP merupakan paket software yang didalamnya sudah terkandung *Web Server Apache, database MySQL dan PHP Interpreter*.(Agustina, 2018)

2.12 MySQL

Menurut (Sibero, 2013), “MySQL adalah suatu *Relational Database Management System* (RDBMS) yaitu aplikasi sistem yang menjalankan fungsi pengolahan data.” MySQL merupakan salah satu aplikasi DBMS yang sudah banyak oleh para pemrogram aplikasi web (Hidayatullah & Kawistara, 2017) MySQL adalah *Relational Database Management System* (RDBMS) yang berfungsi untuk melakukan pengelolahan data dalam membangun suatu aplikasi web. MySQL menggunakan konsep utama dalam database, yaitu SQL (*Structured Query Language*). SQL merupakan struktur bahasa yang berfungsi untuk berkomunikasi dengan database.

2.13 Metode Pengembangan Sistem

2.13.1 Waterfall

Waterfall merupakan suatu proses pengembangan perangkat lunak yang berurutan yang terus mengalir ke bawah seperti air terjun, yang terdiri dari fase perencanaan, pemodelan, implementasi, pengujian dan pemeliharaan (Pressman, 2012).

1. Perancangan

Perancangan adalah proses pengumpulan kebutuhan secara lengkap yang kemudian dianalisis serta mendefinisikan kebutuhan yang harus dipenuhi oleh sistem yang akan dibangun.

2. Pemodelan

Pemodelan atau desain sistem adalah tahap untuk mendefinisikan tampilan sistem secara keseluruhan dan menentukan alur perangkat lunak dalam membangun sistem.

3. Implementasi

Implementasi adalah tahapan yang mana seluruh desain kemudian diubah menjadi kode program. Kode program yang dihasilkan yang masih berupa modulmodul akan diintegrasikan menjadi sistem yang lengkap.

4. Pengujian

Pengujian adalah tahap penggabungan modul-modul yang kemudian dilakukan pengujian apakah sistem tersebut telah sesuai dengan desain dan tidak ada kesalahan pada setiap fungsi sistem.

5. Pemeliharaan

Pemeliharaan adalah tahap setelah sistem telah diimplementasikan dan dilakukan pengembangan akan dilakukan pemeliharaan serta perbaikan pada sistem apabila terdapat kesalahan pada sistem tersebut.

2.14 *Unified Modeling Language* (UML)

Uml merupakan sebuah bahasa yang menggunakan grafik atau gambar untuk memvisualisasikan, mespesifikasikan, membangun dan mendokumentasikan dari sebuah sistem pengembangan software berbasis *object-oriented*. (Komang, 2019)

1. Tujuan UML

a. Memberikan model yang siap pakai, bahasa permodelan visual yang ekspresif untuk mengembangkan model dan dimengerti secara umum.

b. Memberikan bahasa permodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa.

c. Menyatukan praktek-praktek yang terdapat dalam permodelan.

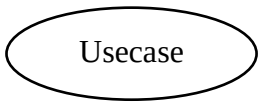
2. Diagram-diagram dalam UML

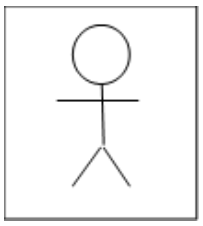
Ada beberapa diagram dalam UML (*Unified Modelling Language*) antara lain:

a. *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 merupakan sebuah pekerjaan tertentu, misalnya Login ke sistem, membuat atau create sebuah daftar belanja dan sebagainya. Seorang aktor adalah entitas manusia atau sebuah mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. Adapun simbol dari use case diagram antara lain :

Tabel 2.1 Komponen *Use Case Diagram*

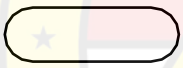
No	Simbol	Keterangan
1	Use Case 	Merupakan sebuah komponen yang menggambarkan seseorang atau sesuatu (seperti perangkat

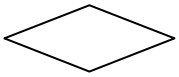
		atau sistem lainnya) yang berinteraksi dengan sistem.
2	Actor 	Use case adalah gambaran fungsionalitas dari suatu sistem, sehingga pengguna sistem paham dan mengerti mengenai kegunaan sistem yang akan dibangun.
3	Asosiasi / Association	Komunikasi antara aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan actor
4	Ekstensi / Extend <<extend>>	Kelakuan yang hanya berjalan di bawah kondisi tertentu seperti menggerakkan alarm.
5	Generalisasi / Generalization	*Generalization* disebut juga inheritance (pewarisan), sebuah elemen dapat merupakan spesialisasi dari element lainnya.
6	Menggunakan / Include / uses <<include>>	Kelakuan yang harus terpenuhi agar sebuah *event* dapat terjadi, di mana pada kondisi ini sebuah use case adalah bagian dari use case lainnya.

b. *Activity Diagram*

Activity diagram menggambarkan *workflow* atau aktivitas dari suatu sistem. Activity diagram menggambarkan aktivitas yang dapat dilakukan oleh suatu sistem.(Sukamto & Shalahuddin, 2018).

Tabel 2.2 Komponen *Activity* Diagram

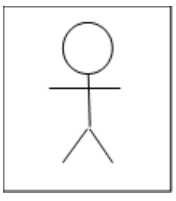
Activity Diagram	**Keterangan**
Start State 	*Start State*, sebagai tanda awal proses dari activity diagram.
State 	*State*, berfungsi menampung event dalam activity diagram.
Activity 	*Activity*, memiliki fungsi yang sama dengan state. Menampung event atau aktivitas pada proses sistem.
State Transition	*State Transition*, berfungsi untuk menunjukkan aliran atau urutan dari event atau aktivitas pada diagram.
Transition to self 	*Transition to self*, berfungsi untuk menunjukkan transisi sebuah event yang mengarah ke event itu sendiri.
Horizontal Synchronization	*Horizontal Synchronization*, berfungsi untuk menyinkronisasikan 2 cabang event yang posisinya horizontal.

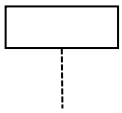
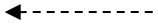
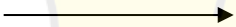
Vertical Synchronization 	*Vertical Synchronization,* berfungsi untuk menyinkronisasikan 2 cabang event yang posisinya vertikal.
Decision 	*Decision,* digunakan ketika terjadi pemilihan 2 kondisi event pada diagram.
End State 	*End State,* sebagai tanda akhir dari activity diagram.

c. Sequence Diagram.

Sequence diagram menggambarkan tingkah laku objek pada use case dengan mendeskripsikan waktu hidup suatu objek dan pesan yang dikirim atau diterima antar objek tersebut. Dalam menggambar sequence diagram perlu diketahui objek yang terlibat pada suatu use case beserta metode yang dimiliki kelas yang didefinisikan menjadi objek tersebut (Sukamto & Shalahuddin, 2018).

Tabel 2.3 Komponen *Sequence* Diagram

Sequence Diagram	Keterangan
Actor 	*Actor,* menggambarkan seseorang atau sesuatu (seperti perangkat atau sistem lain) yang berinteraksi dengan sistem.

Boxes 	*Boxes,* sebuah kontak yang tampil pada posisi paling atas diagram, yang mewakili object, use case, class, dan actor.
Return Massage 	*Return Massage,* menggambarkan pesan atau hubungan antara obyek yang menunjukkan urutan kejadian yang terjadi.
Lifeline 	*Lifeline,* eksekusi obyek selama sequence (message dikirim atau diterima dan aktifasinya)
Message to Self 	*Message to Self,* menggambarkan pesan atau hubungan obyek itu sendiri yang menunjukkan urutan kejadian yang terjadi
Object Message 	*Object Message,* menggambarkan pesan atau hubungan antar obyek yang menunjukkan urutan kejadian yang terjadi.