BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terdahulu

Beradasarkan penelitian yang dilakukan Sri Andayani dan Endah Wulan Perwitasari (2014) mengenai Penentuan Rute Terpendek Pengambilan Sampah di Kota Merauke Menggunakan Algoritma Dijkstra menyampaikan bahwa Kombinasi antara algoritma Dijkstra dengan metode saving heuristic dapat diimplementasikan dengan baik dan dapat menghasilkan rute pengambilan pengambilan sampah di kota Merauke.

Berdasarkan penelitian yang dilakukan Yudhi Purwananto, dkk (2005) mengenai Implementasi Dan Analisis Algoritma Pencarian Rute Terpendek Di Kota Surabaya menyampaikan bahwa hasil rute terpendek yang diperiksa, kinerja algoritma dijkstra lebih baik daripada dua algoritma penghitungan rute terpendek yang lainnya.

Berdasarkan penelitian yang dilakukan Fitria, Apri Triansyah (2013) mengenai Implementasi Algoritma Dijkstra Dalam Aplikasi Untuk Menentukan Lintasan Terpendek Jalan Darat Antar Kota Di Sumatera Bagian Selatan menyampaikan bahwa Algoritma dijkstra dapat digunakan untuk mencari rute terpendek secara optimal.

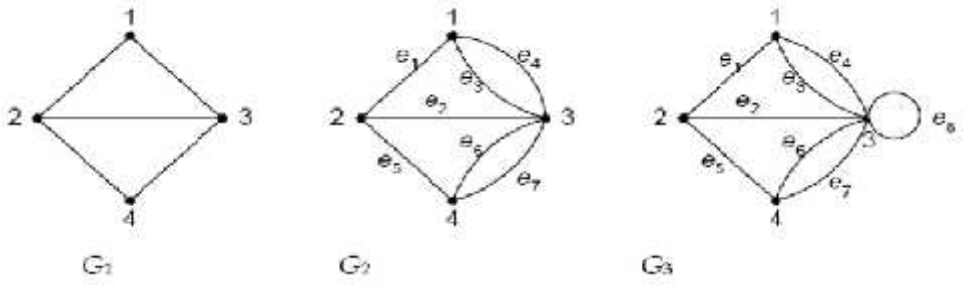
Berdasarkan penelitian yang dilakukan Diana Okta Pugas, dkk (2011) mengenai Pencarian Rute Terpendek Menggunakan Algoritma Dijkstra dan Astar (A*) pada SIG Berbasis Web untuk Pemetaan Pariwisata Kota Sawahlunto

menyampaikan bahwa Aplikasi ini berhasil menemukan rute terpendek antar objek wisata yang ada di Kota Sawahlunto yaitu sebanyak 12 objek wisata menggunakan algoritma Dijkstra dan Astar.

Berdasarkan penelitian yang dilakukan Yogi Primadasa (2015) mengenai Pencarian Rute Terpendek Menggunakan Algoritma Dijkstra Pada Sig Berbasis Web Untuk Distribusi Minuman menyampaikan bahwa Aplikasi Sistem Informasi Geografis yang berbasis web ini mampu menyalurkan informasi untuk PT. Distribusi Coca Cola tentang jalur mana yang bisa ditempuh untuk mempercepat proses pendistribusian dengan mengikuti jalur rute terpendek menggunakan algoritma Dijkstra yang telah dihasilkan.

II.2. Teori Graf

Graf didefinisikan sebagai pasangan himpunan (V, E), ditulis dengan notasi G = (V, E), yang dalam hal ini V adalah himpunan tidak kosong dari simpul-simpul (vertices atau node) dan E adalah himpunan sisi (edges atau arcs) yang menghubungkan sepasang simpul . Simpul pada graf dapat dinomori dengan huruf, seperti a, b, c...dst, dengan bilangan asli 1, 2, 3...dst, atau gabungan keduanya. Sedangkan sisi yang menghubungkan simpul u dengan simpul v dinyatakan dengan pasangan (u, v) atau dinyatakan dengan lambang e1, e2....en dengan kata lain, jika e adalah sisi yang menghubungkan simpul u dengan simpul v, maka e dapat ditulis sebagai e = (u, v). Secara geometri graf digambarkan sebagai sekumpulan noktah (simpul) di dalam bidang dwimatra yang dihubungkan dengan sekumpulan garis (sisi).



Gambar II.1. (G1) graf sederhana, (G2) multigraf, dan (G3) multigraf

(*Sumber : Fitria, Apri Triansyah, 2013*)

Gambar II.1 memperlihatkan tiga buah graf, G1, G2 dan G3 . G1 adalah graf dengan himpunan simpul V dan himpunan sisi E adalah:

V = {1, 2, 3, 4}

E = {(1, 2), (1, 3), (2, 3), (2, 4), (3, 4)}

G2 adalah graf dengan himpunan simpul V dan himpunan sisi E adalah:

V = {1, 2, 3, 4}

E = {(1, 2), (2, 3), (1, 3), (1, 3), (2, 4), (3, 4), (3, 4)}

= {e1, e2, e3, e4, e5, e6, e7}

G3adalah graf dengan himpunan simpul V dan himpunan sisi E adalah:

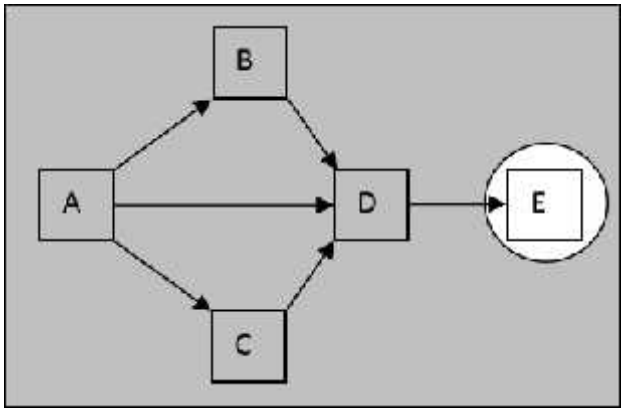
V = {1, 2, 3, 4}

E = {(1, 2), (2, 3), (1, 3), (1, 3), (2, 4), (3, 4), (3, 4), (3, 3)}

= {e1, e2, e3, e4, e5, e6, e7, e8}

Pada G2, sisi e3 = (1, 3) dan sisi e4= (1, 3) dinamakan sisi-ganda (*multiple edges* atau *parallel edges*) karena kedua sisi ini menghubungi dua buah simpul yang sama, yaitu simpul 1 dan simpul 3. Pada G3 , sisi e8 = (3, 3) dinamakan gelang atau kalang (*loop*) karena ia berawal dan berakhir pada simpul yang sama (Fitria, Apri Triansyah, 2013).

II.3. Rute Terpendek



Gambar II.2. Proses penelusuran

(*Sumber : Feddy Setio Pribadi & Anggraini Mulwinda, 2010*)

Perjalanan dengan mempertimbangkan penelusuran rute terpendek hanya mempertimbangkan jarak atau cost antar kota yang harus dilalui. Seperti ditunjukan pada Gambar II.2 Untuk menuju ke kota E dari kota asal yaitu kota A, algoritma pencarian yang efektif akan membandingkan jarak yang harus ditempuh jika melalui kota B kemudian baru ke kota E dengan jarak yang dilalui jika perjalanan dilakukan dari kota A lansung menuju ke kota E.

Pencarian rute terpendek hanya sematamata mempertimbangkan jarak terpendek yang harus dilalui, pada kasus Gambar 3 Suatu metode bisa jadi akan

memilih menelusuri kota A ke kota D untuk mencapai tujauan akhir kota E, karena dengan pertimbangan perbandingan jarak tempuh dari kota A ke kota B lebih dekat daripada jarak yang harus ditempuh dari kota A ke kota D (Feddy Setio Pribadi, Anggraini Mulwinda, 2010).

II.4. Android

Android adalah sistem operasi untuk telepon seluler yang berbasis Linux. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri yang digunakan oleh bermacam peranti bergerak. Untuk mengembangkan Android, maka dibentuklah *Open Handset Alliance*, konsorsium dari 34 perusahaan peranti keras, peranti lunak dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile dan Nvidia.

Pada Juli 2000, Google bekerjasama dengan Android Inc., perusahaan yang berada di Palo Alto, California Amerika Serikat. Di perusahaan Google, tim yang dipimpin oleh Rubin bertugas mengembangkan program perangkat seluler yang didukung oleh kernel Linux. September 2007, sebuah studi melaporkan bahwa Google mengajukan hak paten aplikasi telepon seluler (Google mengenalkan Nexus One, salah satu jenis *smartphone* produksi HTC *Corporation* yang menggunakan Android pada sistem operasinya). Seiring pembentukan *Open Handset Alliance*, OHA mengumumkan produk perdana mereka yaitu Android sebagai perangkat mobile yang merupakan modifikasi kernel Linux 2.6. Sejak Android dirilis, telah dilakukan berbagai pembaruan berupa perbaikan bug dan penambahan fitur baru. Telepon pertama yang menggunakan sistem operasi

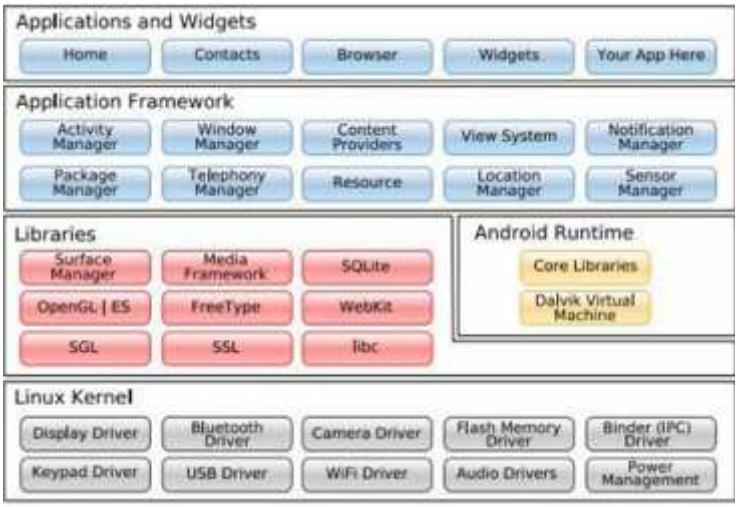
Android adalah HTC Dream, yang dirilis pada 22 Oktober 2008. Pada quartal ke-3 tahun 2010, jumlah penjualan handset berbasis Android mencapai 20,5 juta unit (Murtiwiyati dan Glenn Lauren, 2013).

II.4.1. Fitur

Fitur yang tersedia sampai saat ini pada Android yaitu:

a. *Framework* aplikasi, memungkinkan daur ulang dan penggantian komponen.
b. *Browser* terintegrasi berbasis *engine Open Source WebKit* yang juga digunakan di *browser* IPhone dan Nokia S60v3.
c. Rancangan *handset* p*latform* disesuaikan dengan kebutuhan VGA (*Video Graphics Adapter*) yang lebih besar, library grafik 2D dan 3D yang berdasarkan pada spesifikasi OpenGL ES 1.0 serta layout smartphone yang tradisional.
d. Konektivitas. Android mendukung berbagai teknologi konektivitas seperti GSM (*Global System for Mobile Communications*) EDGE (*Enhanced Data rates for GSM Evolution), CDMA (Code Division Multiple Access*), EV-DO (*Evolution-Data Optimized), UMTS (Universal Mobile Telecommunications System*), *Bluetooth* dan Wi-Fi (*Wireless Fidelity*).
e. Pesan. Android mendukung pengiriman pesan dalam bentuk SMS (*Short Message Service*) dan MMS (*Multimedia Messaging Service*) (M.Rofiq, Riza Fathul Uzzy, 2014).

II.4.2. Arsitektur



Gambar II.3. Arsitektur
(Sariyun Naja Anwar, dkk, 2015)

1. *Applications and Widgets Applications*

   Applications and Widgets Applications ini adalah layer dimana berhubungan dengan aplikasi saja, di mana biasanya download aplikasi dijalankan kemudian dilakukan instalasi dan jalankan aplikasi tersebut

2. *Applications Framework*

   Applications frameworks ini adalah layer di mana para pembuat aplikasi melakukan pengembangan/pembuatan aplikasi yang akan dijalankan di sistem operasi Android, karena pada layer inilah aplikasi dapat dirancang dan dibuat, seperti contectproviders yang berupa sms dan panggilan telepon.

3. *Libraries*

   Libraries ini adalah layer di mana fitur-fitur Android berada, biasanya para pembuat aplikasi mengakses libraries untuk menjalankan aplikasinya. Berjalan

di atas kernel, Layer ini meliputi berbagai *library* C/C++ inti seperti Libc dan SSL.

4. *Android Run Time*

   Android Run Time Layer yang membuat aplikasi Android dapat dijalankan dimana dalam prosesnya menggunakan Implementasi Linux.

5. *Linux Kernel*

   Linux Kernel adalah layer di mana inti dari operating system dari Android itu berada. Berisi file-file system yang mengatur sistem *processing, memory, resource, drivers*, dan sistem-sistem operasi android lainnya. *Linux kernel* yang digunakan android adalah *linux kernel release* 2.6 (Murtiwiyati dan Glenn Lauren, 2013).

II.5. Java

II.5.1. Pengertian Java

Java pertama kali diluncurkan sebagai bahasa pemrograman umum (*general purpose programming language*) dengan kelebihan dia bisa dijalankan di *web browser* sebagai *applet*. Sejak awal, para pembuat Java telah menanamkan visi mereka ke dalam Java untuk *small embedded customer device*) seperti TV, telepon, radio, dan sebagainya supaya dapat berkomunikasi satu sama lain. Langkah pertama yang diambil oleh Sun Micro sistem adalah dengan membuat JVM (*JavaVirtual machine*) yang kemudian diimplementasikan dalam bentuk JRE (*Java Runtime Environment*). JVM adalah lingkungan tempat eksekusi program Java berlangsung dimana para obyek saling berinteraksi satu dengan yang lainnya. *Virtual machine* inilah yang menyebabkan Java mempunyai

kemampuan penanganan memori yang lebih baik, keamanan yang lebih tinggi serta portabilitas yang besar. Apabila kita hanya ingin menjalankan program Java, maka kita cukup memiliki JRE saja.Tapi seandainya kita ingin mengembangkan perangkat lunak sendiri, JRE saja tidak cukup.

Untuk lebih meningkatkan produktivitas pengembang perangkat lunak, Sun juga meluncurkan SDK (*Standard Development Kit*) yang berisi kakas dan API untuk membuat program aplikasi berbasis Java. Pada tahun 1999 Sun meluncurkan J2EE (*Java 2 Enterprise Edition*) sebagai *framework* untuk membuat aplikasi enterprise berskala besar. Pada tahun 2001, Sun meluncurkan J2ME yang kelak menjadi salah satu standar pemrograman di dalam PDA maupun *handphone* (Warno, 2011).

II.5.2. Arsitektur Java

Arsitekturnya yang kokoh dan pemrograman yang aman. Dalam Java program yang kita buat tidak mudah untuk "*hang*" karena konflik pada memori biasanya diselesaikan dengan mengumpulkan obyek-obyek yang sudah tak terpakai lagi secara otomatis oleh garbage collector. Penanganan kesalahan juga dipermudah dalam Java dengan konsep *Exception*.

1. Bukan sekedar bahasa tapi juga *platform* sekaligus arsitektur. Java mempunyai portabilitas yang sangat tinggi. Ia dapat berada pada *smartcard*, *pager*, POS (*Point of Service*), *handphone*, PDA, *palm*, TV, *Embedded device* (PLC, *micro controller*), laptop, pc, dan bahkan server). Menyadari akan hal ini Sun membagi arsitektur Java membagi tiga bagian, yaitu:

a. *Enterprise Java* (J2EE) untuk aplikasi berbasis web, aplikasi sistem tersebar dengan beraneka ragam klien dengan kompleksitas yang tinggi. Merupakan superset dari Standar Java.
b. *Standard Java* (J2SE), ini adalah yang biasa kita kenal sebagai bahasa Java, dan merupakan fokus kita sekarang.
c. *Micro Java* (J2ME) merupakan subset dari J2SE dan salah satu aplikasinya yang banyak dipakai adalah untuk *wireless device*/*mobile device*.

2. Program Java dijalankan menggunakan interpreter melalui *Java Virtual machine* (JVM). Hal ini menyebabkan *source code Java* yang telah dikompilasi menjadi Java *bytecodes* dapat dijalankan pada *platform* yang berbeda-beda.
3. Fitur-fitur utama yang lain:
   a. Mendukung *multithreading*.
   b. Selalu memeriksa tipe obyek pada saat *runtime*.
   c. Mempunyai *automatic garbage collection* untuk membersihkan obyek yang tidak terpakai dari memori.
   d. Mendukung *exception* sebagai salah satu cara penanganan kesalahan (Warno, 2011).

II.6. UML (*Unified Modelling Language*)

Menurut Gata (2013 : 4) Hasil pemodelan pada OOAD terdokumentasikan dalam bentuk *Unified Modeling Language* (UML). UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak.

UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem.

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

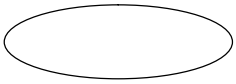
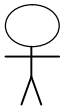
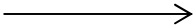
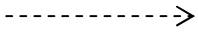
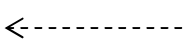
1. *Use Case Diagram*

Use case diagram adalah proses pemodelan fungsi-fungsi sistem dalam konteks pristiwa-pristiwa bisnis, siapa yang mengawali dan bagaimana sistem itu memproses hal tersebut.

Usecase diagram memperlihatkan hubungan diantara aktor dan usecase. Aktor merepresentasaikan seorang user atau sub sistem lain yang akan berinteraksi dengan sistem. Sedangkan usecase merupakan urutan kejadian yang menggambarkan interaksi antara user dengan sistem fungsionalitas sistem didefinisikan ke dalam usecase dari sudut eksternal sistem yang berguna untuk uji

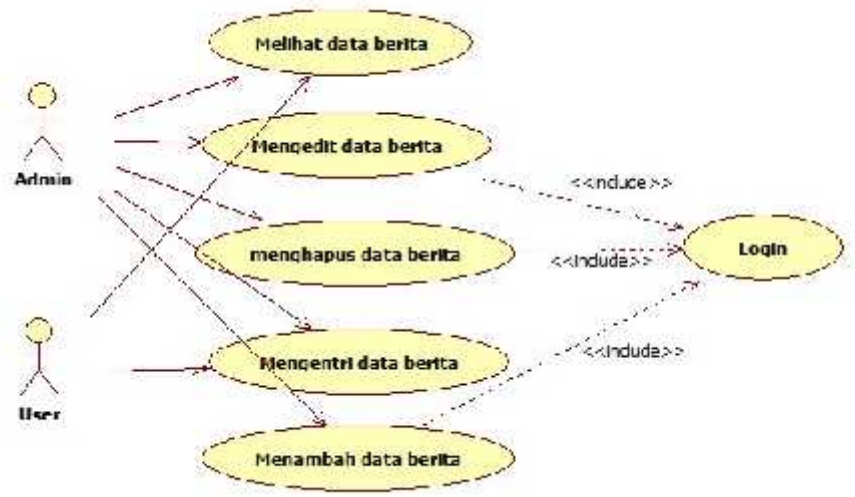
kelayakan fungsionalitas. Simbol-simbol yang digunakan dalam *use case* diagram, yaitu :

Tabel II.1. Simbol *Use Case*

Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan 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 control 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 mengidikasikan aliran data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan 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: Gata ; 2013 : 4)

Gambar II.5 menampilkan usecase diagram dalam UML :

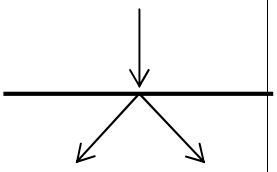
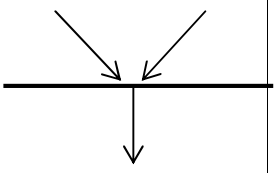
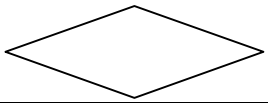


Gambar II.4 ***Use Case Diagram***
(*Sumber: Gata ; 2013 : 4)*

2. *Activity Diagram*

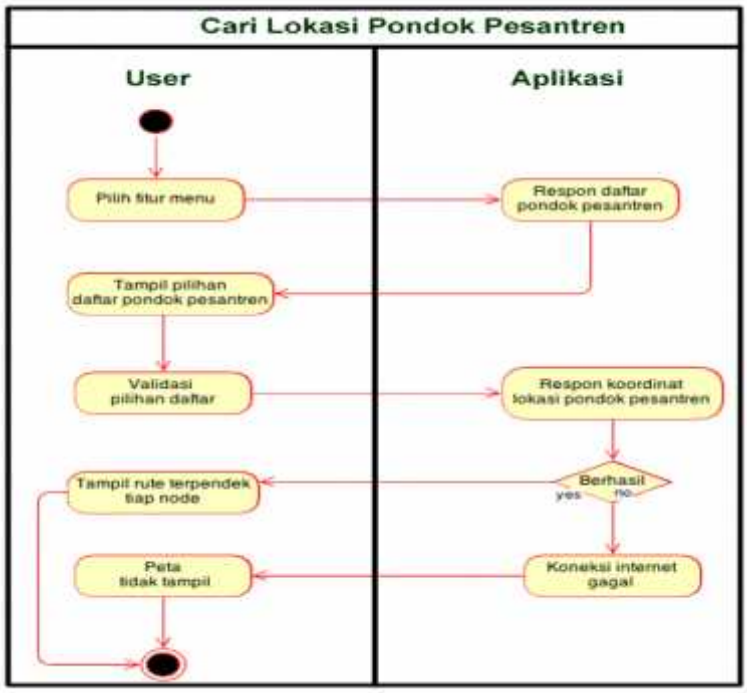
Diagram ini menjelaskan alur kerja sebuah sistem. *Activity diagram* mirip dengan *state diagram* karena sejumlah aktivitas menggambarkan keadaan suatu proses dengan memperlihatkan urutan aktivitas yang dijalankan baik berupa pilihan maupun paralel. Diagram ini juga berguna untuk menganalisa sebuah usecase dengan menggambarkan aksi-aksi yang diperlukan dan kapan aksi-aksi tersebut dijalankan, menjelaskan urutan algoritma yang komplek dan memodelkan sejumlah aplikasi dengan proses paralel. Simbol-simbol yang digunakan dalam *activity diagram*, yaitu:

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 parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.*
	*Join (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.*
	*Decision Points, menggambarkan pilihan untuk pengambilan keputusan, true, false.*
	*Swimlane, pembagian activity diagram untuk menunjukkan siapa melakukan apa.*

(Sumber: Gata ; 2013 : 6)

Gambar II.6 menampilkan activity diagram dalam UML :



Gambar II.5 ***Activity diagram***
(*Sumber: Gata ; 2013 : 6)*

3. *Class Diagram*

Digunakan untuk menggambarkan tipe-tipe objek dan hubungannya dalam sebuah sistem. *Class* diagram memodelkan struktur class dan isinya dengan menggunakan elemen-elemen model seperti *class*, *package* dan objek. Class terdiri dari tiga bagian yaitu nama *class*, atribut dan operasi. *Class* yang didefinisikan secara global dapat diakses oleh objek diluar *class* tersebut.

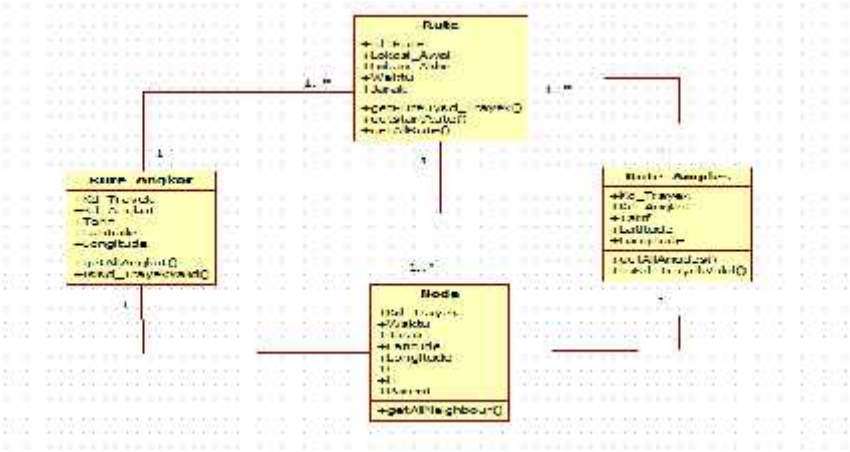
Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau keberagamanan.

Tabel II.3. ***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: Gata ; 2013 : 8)*

Gambar II.7 menampilkan class diagram dalam UML :



Gambar II.6. ***Class diagram***
(*Sumber: Gata ; 2013 : 8)*

Keterangan :

1. Nama kelas ditulis dalam huruf tebal dan diletakkan di tengah-tengah. Nama diambil dari domain permasalahan dan harus sejelas mungkin. Oleh karena itu nama kelas haruslah berupa kata benda.
2. Attribut kelas menggambarkan karakteristik dari objek. Atribut kelas yang benar adalah yang dapat mencakup informasi-informasi yang dilukiskan

dan mengenali instance tertentu dari kelas. Tipe dari atribut dapat berupa primitif atribut atau tipe lainnya.

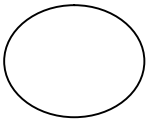
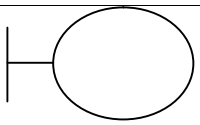
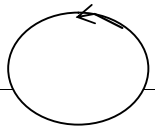
3. Operasi digunakan untuk memanipulasi atribut atau menjalankan aksi-aksi. Operasi biasanya disebut dengan fungsi, tetapi mereka terdapat didalam kelas dan dapat diaplikasikan hanya pada objek dalam kelas tersebut.

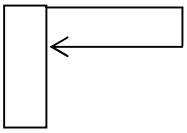
4. *Sequence Diagram*

Diagram ini menjelaskan bagaimana objek berinteraksi dengan lainnya dengan cara mengirim dan menerima pesan. Sequence diagram memiliki dua sumbu: sumbu vertikal dan sumbu horizontal. Sumbu vertikal putus - putus merepre sentasikan"*lifetime*"objek dan sumbu horizontal menunjukan sekumpulan objek.

Sekumpulan objek diagram ini juga menyatakan interaksi khusus diantara objek yang terjadi pada beberapa tempat selama fungsi tertentu dijalankan. Komunikasi diantara objek direpresentasikan dengan garis horizontal disertai dengan nama operasinya. Simbol-simbol yang digunakan dalam *sequence diagram*, yaitu :

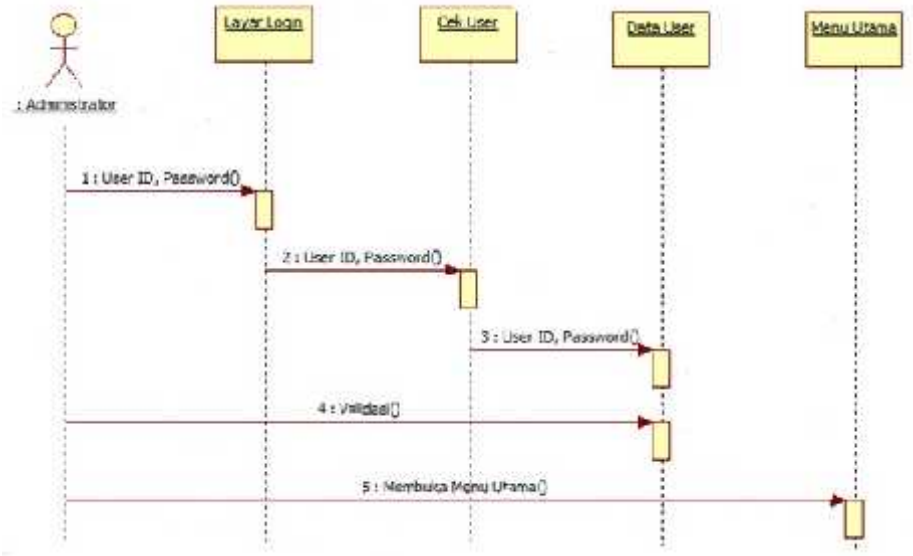
Tabel II.4. Simbol *Sequence Diagram*

Gambar	Keterangan
	*Entity Class*, 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 aplikasi 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: Gata ; 2013 : 7)*

Gambar II.8 menampilkan sequence diagram dalam UML:



Gambar II.7 *Sequence Diagram*
(*Sumber: Gata ; 2013 : 7)*

II.7. Basis Data (*Data Base*)

Basis Data (*database*) memiliki peran yang sangat penting dalam perusahaan. Informasi dapat diperoleh dengan cepat berkat data yang mendasarinya telah disimpan dalam basis data. Pertama, sistem akan memvalidasi keabsahan pemilik kartu dengan memeriksa password yang diberikan oleh orang tersebut. Dalam hal ini, *password* yang diketikan akan dicocokkan dengan *password* pada basis data. Jika sama, langkah berikutnya akan dilaksanakan, yaitu memeriksa saldo uang yang tercatat di basis data terhadap jumlah uang yang diambil. Jika memenuhi syarat, uang akan dikeluarkan oleh mesin.

Sejauh ini basis data tidak hanya berguna pada tataran perusahaan, melainkan juga untuk keperluan pribadi. Dengan menggunakan perangkat lunak basis data seperti *Microsoft Access*, seseorang dapat mengelola data yang menjadi urusan pribadi, seperti data telepon dan data belanja bulanan, dan jika diperlukan segala informasi dapat diperoleh dengan mudah dan cepat.

Basis Data adalah suatu pengorganisasian sekumpulan data yang saling terkait sehingga memudahkan aktivitas untuk memperoleh informasi. Sebagai contoh, basis data akademis mengandung tabel-tabel yang berhubungan dengan data mahasiswa, data jurusan, data matakuliah, data pengambilan mata kuliah pada suatu semester, dan data nilai yang diperoleh mahasiswa (Dimas Radityo Satrio Nugroho, dkk, 2013).

II.7.1 SqLite

SQLite itu merupakan sebuah Database yang bersifat ACID-*compliant* dan memiliki ukuran pustaka kode yang relatif kecil, ditulis dalam bahasa C. SQLite merupakan proyek yang bersifat public domain yang dikerjakan oleh D. Richard Hipp. SQLite adalah sebuah *open source database* yang telah ada cukup lama, cukup stabil, dan sangat terkenal pada perangkat kecil, termasuk Android. Android menyediakan database relasional yang ringan untuk setiap aplikasi menggunakan SQLite. Karena SQLite menggunakan antarmuka SQL, cukup mudah untuk digunakan orang orang dengan pengalaman lain yang berbasis databases. Terdapat beberapa alasan mengapa SQLite sangat cocok untuk pengembangan aplikasi *Android*, yaitu:

Database dengan konfigurasi nol. Artinya tidak ada konfigurasi database untuk para developer. Ini membuatnya relatif mudah digunakan. Tidak memiliki server. Tidak ada proses database SQLite yang berjalan. Pada dasarnya satu set libraries menyediakan *fungsionalitas database*. *Single-file database* ini membuat keamanan database secara langsung. *Open source* hal ini membuat *developer* mudah dalam pengembangan aplikasi (Dimas Radityo Satrio Nugroho, dkk, 2013).

II.8. *Google Maps*

Google Maps merupakan layanan dari google yang mempermudah pengunanya untuk melakukan kemampuan pemetaan untuk aplikasi yang dibuat. Sedangkan *Google Maps* API memungkinkan pengembangan untuk

mengintegrasikan *Google Maps* ke dalam situs web. Dengan menggunakan *Google Maps* API memungkinkan untuk menanamkan situs *Google Maps* ke dalam situs eksternal, di mana situs data tertentu dapat dilakukan *overlay*. Meskipun pada awalnya hanya *JavaScript* API, API *Maps* sejak diperluas untuk menyertakan sebuah API untuk Adobe Flash aplikasi, layanan untuk mengambil gambar peta statis, dan layanan web untuk melakukan *geocoding*, menghasilkan petunjuk arah mengemudi, dan mendapatkan profil elevasi.

Kelas kunci dalam perpustakaan *Maps* adalah *MapView*, sebuah subclass dari *ViewGroup* dalam standar perpustakaan Android. Sebuah *MapView* menampilkan peta dengan data yang diperoleh dari layanan *Google Maps*. Bila *MapView* memiliki fokus, dapat menangkap tombol yang ditekan dan gerakan sentuh untuk pan dan zoom peta secara otomatis, termasuk penanganan permintaan jaringan untuk ubin peta tambahan.

Secara umum, kelas *MapView* menyediakan pembungkus di *Google Maps* API yang memungkinkan aplikasi tersebut memanipulasi data *Google Maps* melalui metode kelas, dan itu memungkinkan dikerjakan dengan data *Maps* seperti jenis lain *Views*.*Google* API pengaya menyediakan perpustakaan *Maps* untuk sehingga dapat mengembangkan, membangun, dan menjalankan aplikasi berbasis peta di SDK Android, dengan akses penuh ke data *Google Maps* (Faya Hahdia dan Fifin Noviyanto, 2013).

II.8.1. Pencarian Koordinat dengan *Google Earth*

Koordinat untuk sebuah konten channel yang berupa *Location Based Services* (LBS) merupakan hal yang sangat penting. Koordinat memiliki peranan penting untuk menampilkan posisi objek. Adapun cara untuk melakukan pencarian koordinat sebuah objek, yaitu dengan menggunakan *Google Earth*.

Satelit merupakan benda yang mengorbit benda lain dengan periode revolusi dan rotasi tertentu. Ada berbagai macam jenis satelit, di antaranya yaitu satelit alam dan satelit buatan. Dalam melakukan pencarian koordinat ini penulis menggunakan satelit navigasi.

Sebuah perangkat mobile memerlukan koneksi internet untuk mendapatkan sinyal langsung ke satelit dalam pencarian data GPS pada permukaan bumi. Dibutuhkan adanya sambungan ke internet untuk dapat menggunakan aplikasi *Google Earth* (Faya Hahdia dan Fifin Noviyanto, 2013).