BAB 1

PENDAHULUAN

1.1 Latar Belakang

Bengkel merupakan salah satu usaha dalam bidang jasa seperti perbaikan atau perawatan kendaraan. Hampir setiap orang saat ini telah memiliki kendaraan baik itu sepeda motor ataupun mobil. Namun ketika berpergian jauh tidak jarang kendaraan yang di kemudikan mengalami kerusakan baik itu berupa bocor ban dan kerusakan lainnya. Ketika mengalami hal tersebut biasanya seseorang akan mengalami kesulitan untuk mencari bengkel terdekat.

Saat ini perkembangan teknologi informasi dalam mengakses internet telah dapat digunakan dengan menggunakan *Mobile*, sehingga mempermudah pengguna jalan dalam mencari keberadaan bengkel terdekat hanya melalui *Mobile*. Banyak Algoritma yang bisa digunakan untuk menentukan rute terpendek, salah satu Algoritma yang dapat di gunakan untuk mengembangkan aplikasi pencarian bengkel dengan rute terdekat adalah Algoritma *Dijkstra*.

Pasir Pengaraian merupakan Ibu kota Kabupaten Rokan Hulu Provinsi Riau dan terbentuk sebagai hasil pemekaran dari Kabupaten Kampar sejak tahun 1999, berdasarkan UU No. 53 tahun 1999 Jo Undang-Undang No. 11 tahun 2004 dan Keputusan Menteri Dalam Negeri Nomor 75 tahun 1999 kemudian diubah terakhir dengan UU No. 34 tahun 2008 [5]. Penelitian ini akan dilakukan di Kota Pasir Pengaraian sebagai tempat pengujian Aplikasi.

Oleh karena itu, penulis mengimplementasikan algoritma *Dijkstra* ke dalam penelitian ini dengan judul Aplikasi Pencarian Bengkel dengan Rute Terpendek Menggunakan Algoritma *Dijkstra* Studi Kasus: Kota Pasir Pengaraian.

1.2 Rumusan Masalah

Berdasarkan latar belakang, maka hal yang menjadi pokok permasalahan adalah bagaimanakah aplikasi pencarian bengkel dengan rute terpendek menggunakan algoritma *Dijkstra*?.

1.3 Batasan Masalah

Agar dalam pengerjaan tugas akhir ini dapat lebih terarah, maka penulis membuat batasan-batasan sebagai berikut:

1. Aplikasi di buat menggunakan *platform* android sehingga hanya dapat digunakan oleh perangkat *mobile*.
2. Penelitian ini dilakukan di Pasir Pengaraian, Kabupaten Rokan Hulu.

1.4 Tujuan Penelitian dan Manfaat Penelitian

1.4.1 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk membuat aplikasi pencarian bengkel dengan rute terpendek meggunakan algoritma *Dijkstra* berbasis *Android*.

1.4.2 Manfaat Penelitian

Adapun manfaat penelitian adalah sebagai berikut:

1. Mengimplementasikan algoritma *Dijkstra* kedalam aplikasi *mobile*.

2. Mempermudah dalam mencari bengkel bagi pengguna jalan.

3. Menjadi media promosi bagi pemilik usaha bengkel.

1.5 Sistematika Penulisan

Berikut merupakan rencana susunan sistematika penulisan laporan Tugas Akhir yang akan dibuat :

BAB 1 PENDAHULUAN

Bab ini berisi penjelasan mengenai latar belakang masalah, rumusan masalah, batasan masalah, tujuan dan sistematika penulisan dari Tugas Akhir yang dibuat.

BAB 2 LANDASAN TEORI

Bab ini membahas tentang teori-teori berhubungan dengan tugas akhir ini.

BAB 3 METODOLOGI PENELITIAN

Bab ini membahas langkah-langkah yang dilaksanakan dalam proses penelitian, yaitu pengamatan pendahuluan dan pengumpulan data, tahapan identifikasi masalah, perumusan masalah, analisa aplikasi, perancangan aplikasi dan implementasi beserta pengujian.

BAB 4 ANALISA DAN PERANCANGAN

Bab ini berisi pembahasan mengenai kebutuhan sistem.

BAB 5 IMPLEMENTASI DAN PENGUJIAN

Bab ini berisi penjelasan mengenai implementasi.

BAB 6 PENUTUP

Bab ini berisi kesimpulan dan saran untuk penelitian selanjutnya.

BAB 2

LANDASAN TEORI

2.1 *Dijkstra*

Pencarian rute terpendek termasuk ke dalam materi teori graf. Algoritma yang sangat terkenal untuk menyelesaikan persoalan ini adalah algoritma *Dijkstra*. Algoritma ini ditemukan oleh seorang ilmuwan komputer berkebangsaan Belanda yang bernama Edsger Dijkstra [4]. Bila *node* dari sebuah graf melambangkan kota-kota dan bobot sisi (*edge weights*) melambangkan jalur antara kota-kota tersebut, maka algoritma *Dijkstra* dapat digunakan untuk menemukan jalur terpendek antara dua kota. Dalam menentukan jalur terpendek dari suatu graf oleh algoritma *dijkstra* akan didapatkan jalur yang terbaik, karena pada waktu penentuan jalur yang akan dipilih, akan dianalisis bobot dari *node* yang belum terpilih, lalu dipilih *node* dengan bobot yang terkecil. Jika ternyata ada bobot yang lebih kecil melalui node tertentu maka bobot akan dapat berubah. Algoritma *dijkstra* akan berhenti ketika semua node sudah terpilih. Sehingga akan ditemukan jalur terpendek dari seluruh node, tidak hanya untuk node dari asal dan node tujuan tertentu saja.

2.2 *Object Oriented Programing* (OOP)

Menurut Rosa dan Shalahuddin [11, h.100] OOP atau pemrograman berorientasi objek adalah satu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data operasi yang

diberlakukan terhadapnya. Rosa dan Shalahuddin menjelaskan juga kelebihan pemrograman berorientasi objek antara lain sebagai berikut ini:

1. Meningkatkan produktivitas
2. Kecepatan pengembangan
3. Kemudahan pemeliharaan
4. Adanya konsistensi
5. Meningkatkan kualitas perangkat lunak

Untuk itu dalam penelitian ini penerapan menggunakan OOP di prioritaskan karena banyak kelebihan dan kemudahan dalam menggunakan pemrograman berorientasi objek karena metode pengembangan dapat menjadikan komponen sebagai sebuah objek.

2.3 PHP: *Hypertext Preprocessor*

Menurut Anhar [1, h.3] PHP adalah singkatan dari *Hypertext Preprocessor* yaitu bahasa pemrograman web *serverside* yang bersifat *open source*. PHP merupakan bahasa pemograman yang berintegrasi dengan HTML dan berada pada *server*. PHP adalah *script* yang menghasilkan halaman *website* yang dinamis. Dinamis berarti yang ditampilkan dibuat saat halaman itu diminta oleh *client*. Mekanisme ini menyebabkan sebuah *website* menjadi lebih mudah memberikan informasi yang diterima *client* selalu yang terbaru / *up to date*.

Sementara menurut MacIntyre [10, h.2] PHP merupakan bahasa skrip, kebanyakan dijalankan di *server*, yang dapat digunakan untuk menghasilkan *Hypertext Markup Language* (HTML) secara dinamis. PHP dijalankan dengan *web server*, biasanya *Apache*, IIS, dan *Ngix* .

Menurut Valade [15, h.16], PHP dapat berkembang sangat cepat karena memiliki banyak kelebihan, antara lain sebagai berikut:

1. Cepat, karena kode PHP tertanam dalam HTML, sehingga memiliki respons yang lebih cepat;

2. Tidak mahal, karena PHP tersedia secara gratis dan bebas digunakan oleh siapa saja;

3. Mudah digunakan, PHP berisi banyak fitur-fitur khusus dan fungsi yang dibutuhkan untuk membuat halaman web dinamis;

4. Dapat berjalan di berbagai sistem operasi, *Windows*, *Linux*, Mac OS, dan turunan dari *Unix*;

5. Dukungan tersedia secara luas, banyak pengguna yang memberikan dukungan secara gratis melalui *email* dan forum;

6. Aman, karena kode PHP tidak terlihat saat dieksekusi dan hanya menghasilkan HTML;

7. Dirancang untuk mendukung *database*, PHP memang dirancang untuk berinteraksi dengan *database*;

8. *Open source*, lisensi *open source* memungkinkan *programmer* untuk memodifikasi *software* PHP agar sesuai dengan kebutuhan.

Dari seluruh penjelasan di atas PHP adalah bahasa pemrograman berbasis *Web* yang berjalan di *server* dan berfungsi untuk menghasilkan sebuah halaman dinamis. PHP dapat juga menghubungkan dengan pemograman *database* sebagai penyimpanan data, salah satunya *database* yang kompatibel dengan PHP adalah MySQL. PHP juga bersifat *open source* sehingga pengguna dapat memodifikasi PHP agar sesuai kebutuhan.

2.4 *Framework* / Kerangka Kerja

Menurut Pratama [12, h.10] *Framework* adalah rangka atau kerangka, arti istilah tersebut dalam dunia pemrograman adalah kumpulan kelas (*class*) dan fungsi (*function, method*) yang disusun secara sistematis berdasarkan kegunaan atau fungsi tertentu untuk mempermudah pembuatan atau pengembangan suatu aplikasi [12].

Pratama [12, h.10] juga menjelaskan bahwa sebagian besar *framework* yang beredar saat ini dibangun berdasarkan konsep *Object-Oriented Programming*. Selain itu banyak manfaat yang didapat saat menggunakan *framework*. *Framework* menawarkan penghematan waktu kerja dalam penulisan kode dan pengaturan berkas-berkas kode. *Programmer* tidak perlu susah payah menulis kode dari awal untuk fungsi-fungsi yang sudah disediakan. Selain itu berkas kode akan tersusun secara sistematis sesuai dengan struktur yang ditawarkan *framework*, dengan demikian akan memberikan kemudahan saat satu *software* harus

dikerjakan oleh banyak orang.

Dari pengertian di atas maka dapat disimpulkan bahwa *framework* merupakan kerangka yang berisi kumpulan *class* dan *function* yang telah di sediakan oleh pengembang atau pembuat *framework* yang disusun secara sistematis agar mempermudah dalam pengembangan suatu aplikasi baik secara individu ataupun kelompok. kekurangan sebuah *framework* adalah *programmer* dituntut untuk mempelajari fungsi yang telah disediakan oleh *framework* untuk menggunakannya secara maksimal.

2.5 *Codeigniter* (CI)

CodeIgniter adalah *toolkit* untuk orang yang membangun aplikasi *web* menggunakan PHP. Tujuannya adalah untuk memungkinkan mengembangkan proyek lebih cepat daripada menulis kode dari awal, CI telah menyediakan kumpulan fungsi yang kaya untuk tugas-tugas yang biasa dibutuhkan, serta antarmuka yang sederhana dan struktur logis untuk mengakses fungsi ini. *CodeIgniter* memungkinkan kreatif fokus pada proyek dengan meminimalkan jumlah kode yang diperlukan untuk tugas yang diberikan [2].

Codeigniter adalah *framework* yang sangat ringan. Sistem inti hanya membutuhkan beberapa fungsi yang sangat kecil. Sangat berbeda dengan banyak kerangka kerja yang membutuhkan sumber daya yang jauh lebih besar. Fungsi tambahan dimuat secara dinamis berdasarkan permintaan, berdasarkan kebutuhan pengembang untuk proses yang diberikan, sehingga sistem dasarnya sangat ramping dan cukup cepat. *CodeIgniter* menggunakan pendekatan *Model-*

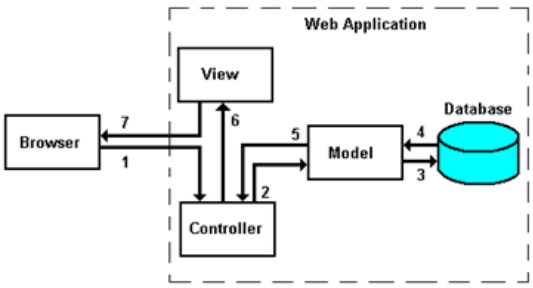
View-Controller, yang memungkinkan pemisahan antara logika dan tampilan.

Oleh karena itu, menggunakan *Codeigniter* dalam penelitian ini membantu pengerjaan aplikasi lebih cepat dan menghemat waktu. dengan dokumentasi yang lengkap *programmer* dapat lebih mudah dalam menggunakan *framework* ini lebih maksimal.

2.6 *Model View Controller* (MVC)

Menurut penjelasan Pratama [12, h.11] Arsitektur MVC muncul sejak tahun 1970 atas pemikiran Prof. Trygve Reenskaug, seorang berkebangsaan Norwegia. Dasar arsitektur ini adalah pemisahan antara logika aplikasi dengan tampilan. Dengan menggunakan pola ini diharapkan dapat meminimalisasi penulisan perintah, sehingga risiko terjadinya *bug* juga *minimal*, serta meningkatkan efisiensi pembangunan suatu aplikasi.

Penjelasan mengenai arsitektur MVC adalah seperti pada Gambar 2.1.



Gambar 2.1: Arsitektur MVC

Pratama [12, n.12] menjelaskan fungsi dari masing-masing bagian adalah sebagai berikut ini:

1. *Model* bertanggung jawab untuk melakukan pengelolaan data dalam basis data, di dalamnya biasa dituliskan perintah untuk mengambil, mengubah, menghapus, dan menambahkan data;

2. *View* merupakan tempat untuk meletakkan apa yang akan ditampilkan di halaman perambah (*browser*), sebuah berkas *view* umumnya berisi kode bahasa pemrograman sisi klien (*client-side scripting*);

3. *Controller* merupakan pengatur utama hubungan antara *model*, *view*, dan juga sumber daya lain yang tersedia, sumber daya ini diperoleh dari kelompok/ tipe kelas yang dapat disebut dengan elemen *framework* CI.

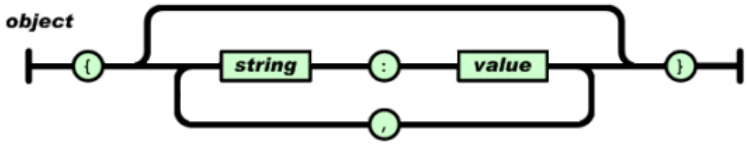
Berdasarkan penjelasan di atas maka dapat diambil kesimpulan bahwa MVC adalah dasar arsitektur pemrograman di mana terdapat pemisahan antara logika, basis data, dan tampilan guna meminimalisasi kesalahan dan juga mempermudah dalam pembagian kerja sama dalam tim.

2.7 *JavaScript Object Notation* (JSON)

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemprograman *JavaScript*, Standar ECMA-262 Edisi ke-3 Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python.

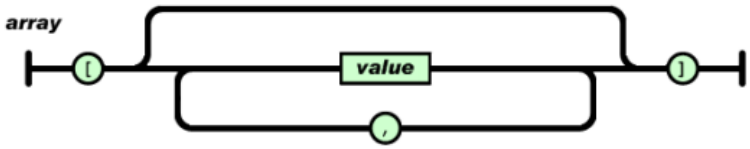
Oleh karena sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data [8].

Objek adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan (kurung kurawal buka) dan diakhiri dengan (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai di-pisahkan oleh , (koma). Dapat di gambarkan pada 2.2 Objek Larik adalah



Gambar 2.2: JSON *Object*

kumpulan nilai yang terurutkan. Larik dimulai dengan [(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma). Dapat di gambarkan pada 2.3 Larik/*Array*



Gambar 2.3: JSON *Array*

2.8 *Android*

Android adalah sebuah sistem operasi untuk perangat *mobile* berbasis linux yang mencakup sistem operasi, *middleware*, dan aplikasi. *Android* menyedi-akan *platform* terbuka bagi pengembang untuk menciptakan aplikasi mereka. awalnya Google Inc membeli Android Inc yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel atau *smartphone*, kemudian untuk

mengembangkan *android* dibentuklah *Open Hardset Alliance*, konsorsium dari 34 perusahaan piranti keras, piranti lunak, dan telekomunikasi, termasuk Google Inc, dan Nvidia [13]. Menurut Kamdar, *Android* memiliki misi membawa kekuatan komputasi ke semua orang. Sebagai sistem operasi global, *Android* telah berkembang menjadi lebih dari 2 miliar perangkat aktif di seluruh dunia, dengan lebih banyak pengguna di India daripada Amerika Serikat [9].

2.9 *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 Studio* merupakan IDE resmi untuk pengembangan aplikasi *Android*. [7]. Android Studio mempermudah dalam pengembangan aplikasi di *Platform Android*, seperti Sistem pengeditan yang fleksibel dan menyediakan *Emulator* tersendiri untuk mencoba pasang dan menjalankan aplikasi yang sedang dikembangkan.

2.10 *Unified Modelling Language* (UML)

Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Dengan menggunakan UML kita dapat membuat model untuk semua je-

nis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan [3].UML mendefinisikan diagram-diagram sebagai berikut:

2.11 *Location Based Service*

Location Based Service (LBS) atau layanan berbasis lokasi adalah layanan informasi yang dapat diakses melalui *mobile device* dengan menggunakan *mobile network*, yang dilengkapi kemampuan untuk memanfaatkan lokasi dari *mobile device* tersebut [14]. LBS memberikan kemungkinan komunikasi dan interaksi dua arah. Selain itu LBS digunakan untuk menggambarkan teknologi yang digunakan untuk menemukan lokasi perangkat atau suatu objek tertentu, seperti menemukan lokasi mesin ATM terdekat atau mengetahui keberadaan teman. *Location Based Service* menggabungkan tiga teknologi sekaligus yaitu *Geographic Information System* (GIS), *Internet Service*, dan *Mobile Device* [13].

2.11.1 *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* merupakan sebuah pekerjaan tertentu, misalnya *login* ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

2.11.2 *Class Diagram*

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 (metoda/fungsi).

2.11.3 *Statechart Diagram*

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu state ke state lainnya) suatu objek pada sistem sebagai akibat dari stimuli yang diterima. Pada umumnya *statechart* diagram menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).

2.11.4 *Activity Diagram*

Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

2.11.5 *Sequence Diagram*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

2.11.6 *Collaboration Diagram*

Collaboration diagram juga menggambarkan interaksi antar objek seperti *Sequence Diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*. Setiap *message* memiliki *sequence number*, di mana *message* dari *level* tertinggi memiliki nomor 1. *Messages* dari *level* yang sama memiliki prefiks yang sama.

2.11.7 *Component Diagram*

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya.

2.11.8 *Deployment Diagram*

Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, *server* atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi *server*, dan hal-hal lain yang bersifat fisikal

BAB 3

METODOLOGI PENELITIAN

Metodologi penelitian adalah langkah dan prosedur yang akan dilakukan dalam pengumpulan data atau informasi guna memecahkan masalah. Metode yang digunakan dalam melaksanakan penelitian merupakan dasar penyusunan rancangan penelitian dan merupakan penjabaran dari metode ilmiah secara umum [6].

3.1 Jenis Penelitian

Tahap ini merupakan tahap pengumpulan pengetahuan dari sumber-sumber seperti buku, internet, dokumen atau publikasi dari berbagai sumber dan perguruan tinggi. Sumber pengetahuan tersebut dijadikan sebagai landasan teori untuk mengembangkan sistem pencarian rute menggunakan alogritma Dijkstra dalam sistem operasi *Android*.

3.2 Wawancara

Pada tahap ini penulis melakukan wawancara dengan pemilik Bengkel Mobil Star di Pematang Berangan untuk memperoleh keterangan mengenai objek penelitian dan berbagai kebutuhan pengguna yang akan menggunakan sistem pencarian bengkel berbasis *android*. Penulis akan meminta keterangan mengenai informasi dan fasilitas bengkel.

3.3 Observasi

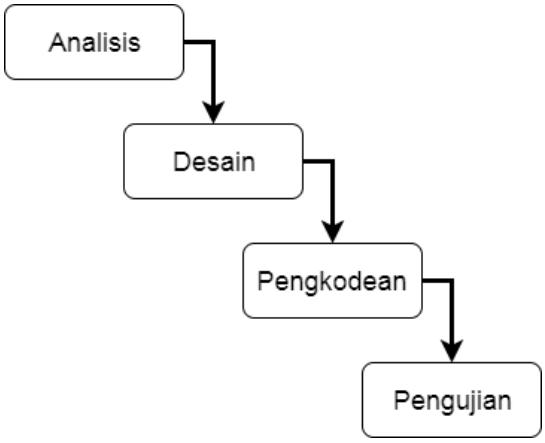
Observasi antara lain pengumpulan data melalui pengamatan dan pencatatan secara langsung pada Bengkel Mobil Star, dengan tujuan untuk memperoleh informasi yang sesuai untuk mendukung data, terutama pada hal-hal yang berkaitan dengan dengan program, sehingga penulis mendapatkan gambaran secara lengkap dan jelas.

3.4 Tempat dan Waktu Penelitian

Penelitian ini dilakukan di Bengkel Mobil Star Desa Pematang Berangan untuk mengumpulkan data yang diperlukan dalam rancangan aplikasi, penelitian ini dilakukan dari bulan Maret 2018 sampai dengan Juni 2018.

3.5 Metode Pengembangan Sistem

Pengembangan aplikasi sistem pencarian bengkel ini akan menggunakan metode *Waterfall*. Model SDLC air terjun (*waterfall*) sering disebut juga model sekuensial linier (*sequiential linier*) atau alur hidup klasik (*classic life cycle*). Model *waterfall* menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengkodean, pengujian dan tahap pendukung (*support*) [11].



Gambar 3.1: *Model Waterfall*

1. Analisis

   Proses menganalisis dan pengumpulan kebutuhan sistem yang sesuai dengan fasilitas dan antar muka (*interface*) yang diperlukan.

2. Desain

   Dalam tahap ini penulis akan merancang desain dan model aplikasi yang akan dikembangkan berdasarkan hasil analisa pada tahap sebelumnya.

3. Kode

   Pengekodean (*coding*) merupakan proses membangun program ke dalam sehingga dapat di terjemahkan ke dalam suatu bahsa yang bisa dimengerti oleh komputer.

4. Test

   Proses Pengujian memastikan bahwa semua program berjalan sesuai dengan fasilitas yang berfungsi dengan benar. Dan memeriksa kesalahan dalam pengkodean yang salah.

3.6 Sarana Pendukung dan Sarana Pengujian

Pada dasarnya setiap penelitian selalu menggunakan sarana pendukung dan sarana pengujian jadi berikut ini penulis akan menjelaskan perangkat dan *software* yang digunakan dalam penelitian.

3.6.1 Perangkat Keras

a.	*Prosessor*	Intel Core i7-3610QM CPU
b.	*Memory* (RAM)	8 GB
c.	*System type*	64-bit Operating System
d.	*Harddisk*	500 GB

3.6.2 Perangkat Lunak

a.	Sistem Operasi	Windows 7
b.	Bahasa Pemograman	Java, PHP, MySQL
c.	*Tool*	Android Studio, Sublime Text
d.	*Web Server*	XAMPP