BAB II
LANDASAN TEORI

2.1 Penelitian Terdahulu

Yanto, dkk., (2017) dalam penelitiannya menggunakan sistem pakar dengan *metode forward chaining* untuk mendiagnosa penyakit pada anak di bawah lima tahun. Dalam penelitiannya dibangun sistem pakar berbasis *rule based* dengan metode *forward chaining,* metode digunakan karena program aplikasi yang dibangun membutuhkan suatu masukan data dari pengguna dan pendekatan yang terdapat dalam MTBS 2009 (Manajemen Terpadu Balita Sakit) berupa inferensi, dimana basis pengetahuan yang terdapat dalam MTBS 2009 diekstrak kedalam sistem pakar. Sistem pakar ini diharapkan dapat mendiagnosa penyakit pada Balita dengan cepat dan tepat. Penelitian dibangun menggunakan bahasa pemrograman *java* berbasis *android* dan *database SQLite. Database SQLite* digunakan untuk menyimpan hasil keluaran aplikasi.

Mulyani dan Restianie (2016) dalam penelitiannya merancang dan membangun aplikasi sistem pakar untuk mendiagnosa penyakit anak (balita) dengan metode *forward chaining*. Metode ini cocok untuk diagnosa awal pada penyakit dengan pelacakan dari gejala-gejala yang diderita. Aplikasi ini diharapkan dapat memberi informasi diagnosa dan cara penanganan secara tepat kepada orang tua mengenai penyakit anak yang sering diderita. Dalam perancangan aplikasi digunakan metode *waterfall* diawali dengan analisis data, perancangan sistem, pengkodean menggunakan *visual basic* 6.0, dan pengujian sistem menggunakan *black box test*. Pengumpulan data melakukan wawancara dengan pakar, hasil wawancara didapatkan data yang digunakan untuk proses perancangan aplikasi.

Salisah, dkk., (2015) dalam penelitiannya sistem pakar penentuan bakat anak dengan metode *forward chaining.* Metode *forward chaining* digunakan sebagai mesin inferensi di sistem pakar yang dibangun karena teknik ini telah sukses digunakan untuk sistem pakar pada berbagai bidang. Sistem ini diharapkan dapat menjadi alat untuk mengidentifikasi bakat anak. Kelompok anak yang digunakan pada penelitian ini adalah anak taman kanak-kanak dengan usia 4-6

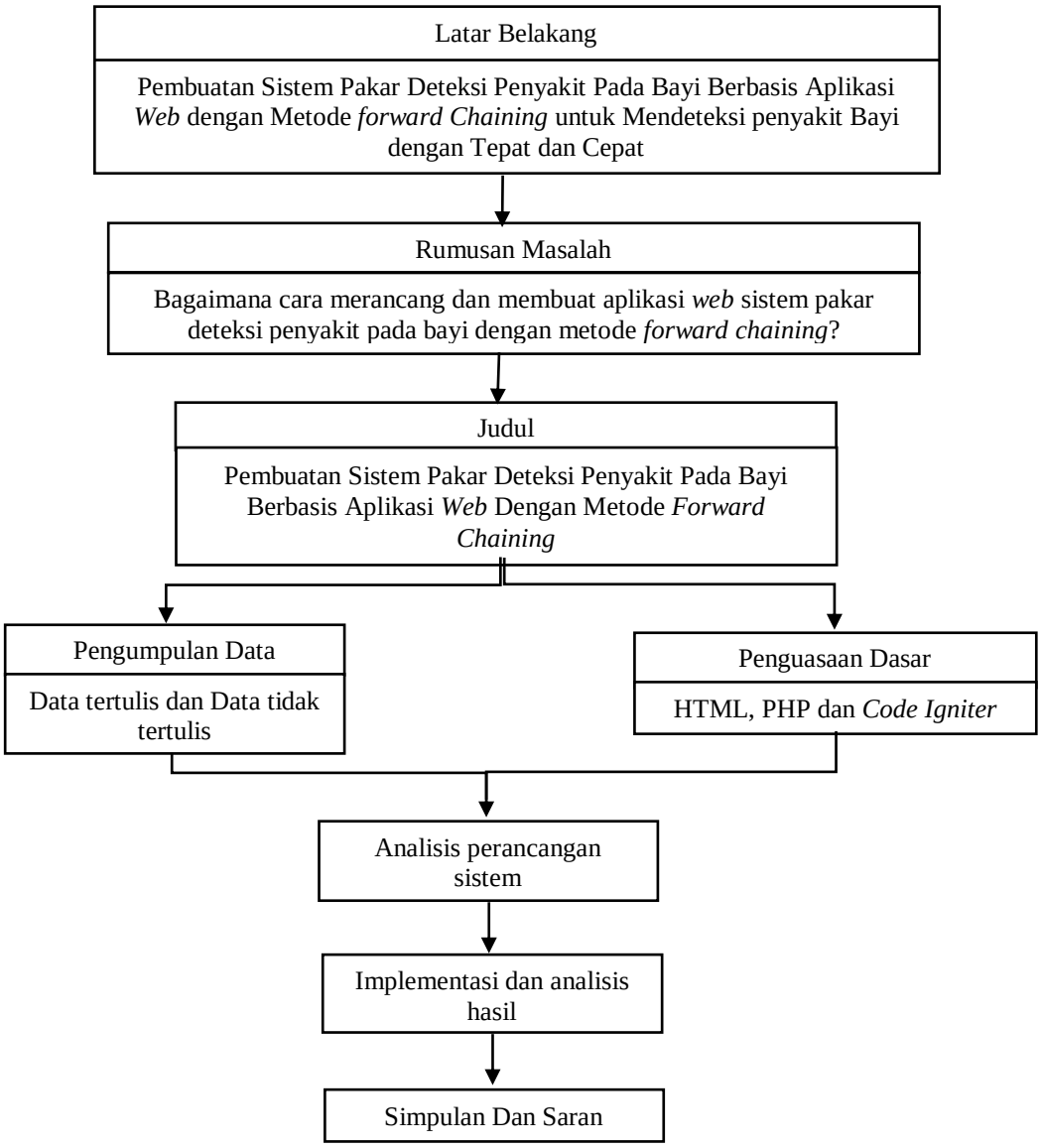
tahun. Kelompok usia tersebut dipilih karena pada usia tersebut merupakan masa yang penting untuk meletakkan dasar bagi seseorang di masa dewasa. Penelitian sistem ini dibagi menjadi tiga tahap: tahap inisialisasi, tahap analisa perancangan, dan tahap implementasi pengujian.

Verina (2015) dalam penelitiannya penerapan metode *forward chaining* untuk mendeteksi penyakit THT, mengunakan metode *forward chaining* untuk mencocokan fakta gejala yang terjadi kemudian menghasilkan suatu kesimpulan atau tujuan. Sistem ini diharapkan dapat memberikan kesimpulan jenis penyakit THT yang diderita. Sistem ini dibuat dengan mengidentifikasikan masalah yaitu memperhatikan gejala penyakit THT dan pemilihan akuisisi pengetahuan. Selanjutnya menganalisa masalah dalam melakukan analisa masalah peneliti menggunakan metode deskriptif. Pada metode ini data yang akan dikumpulkan, disusun, dikelompokkan, dianalisa sehingga diperoleh beberapa gambaran yang jelas pada masalah penelitian. Kemudian menetapkan tujuan, studi literatur dan mengumpulkan data primer yaitu dengan cara interview dan observasi dengan pakar THT.

Trianto (2018) dalam penelitiannya penerapan metode *forward chaining* untuk diagnosa penyakit diare pada anak usia 3-5 tahun berbasis *mobile android* membuat sistem yang digunakan untuk menunjang pelayanan kesehatan terhadap anak khususnya balita. Sistem ini akan membandingkan gejala yang diderita oleh seorang pasien dengan sebuah data tentang penyakit beserta gejalanya. Kemudian hasil dari perbandingan tersebut akan diambil yang memiliki tingkat ketepatan yang paling tingggi. Sehingga telah diketahui diagnosa tentang penyakit seorang pasien sehingga dapat diketahui pula penanganan yang tepat untuk mencegah terjadinya gejala penyakit lebih lanjut.

2.2 Kerangka Pemikiran

Kerangka berpikir adalah penjelasan sementara secara konseptual tentang keterkaitan hubungan pada setiap objek permasalahan berdasarkan teori. Kerangka pemikiran pembuatan sistem pakar deteksi penyakit pada bayi ditunjukan pada Gambar 2.1.



Gambar 2.1 Diagram Kerangka Pemikiran

2.3 Tinjauan Pustaka

2.3.1 Sistem Pakar

Sistem pakar adalah program komputer yang meniru kemampuan beberapa pakar di bidang tertentu dalam memecahkan masalah seperti para pakar tersebut memecahkan masalah dalam bidangnya. Proses peniruan tersebut melibatkan empat hal yaitu: akuisisi pengetahuan, representasi pengetahuan, inferensi pengetahuan, pemindahan pengetahuan ke pengguna (Salisah, dkk., 2015).

Sistem pakar adalah program-program yang memberi saran secara otomatis yang mencoba untuk meniru proses-proses berpikir dan pengetahuan dari ahli-ahli untuk meraih sasaran dari masalah tertentu (Akil, 2017). Komponen-komponen pada sistem pakar saling berhubungan dalam melakukan proses untuk memberikan informasi. Adapun komponen sistem pakar adalah sebagai berikut.

1. Antarmuka pengguna (*user interface*)

User interface merupakan mekanisme yang digunakan oleh pengguna dan sistem pakar untuk berkomunikasi. Sistem pakar menampilkan pertanyaan-pertanyaan yang hanya perlu dijawab oleh pengguna. Pada bagian ini terjadi dialog antar program dan pemakai, yang memungkinkan sistem pakar menerima instruksi dan masukan dari pemakai, juga memberikan informasi hasil kepada pemakai (Listiyono, 2008).

2. Basis pengetahuan (*knowledge base*)

Basis pengetahuan mengandung pengetahuan untuk pemahaman, formulasi, dan penyelesaian masalah. Komponen sistem pakar ini disusun atas dua elemen dasar, yaitu fakta dan aturan. Fakta merupakan informasi tentang obyek dalam area permasalahan tertentu, sedangkan aturan merupakan informasi tentang cara bagaimana memperoleh fakta baru dari fakta yang telah diketahui (Listiyono, 2008).

3. Akuisisi pengetahuan

Akuisisi pengetahuan adalah komulasi, transfer dan transformasi keahlian dalam menyelesaikan masalah dari sumber pengetahuan kedalam program komputer. Dalam tahap ini *knowledge enginer* berusaha menyerap *knowledge* untuk ditransfer

kedalam basis pengetahuan. Pengetahuan diperoleh dari pakar dilengkapi dengan buku, basis data, laporan penelitian dan pengalaman pemakai (Listiyono, 2008).

4. Mesin inferensi

Komponen ini mengandung mekanisme pola pikir dan penalaran yang digunakan oleh pakar dalam menyelesaikan masalah. Mesininferensi adalah program komputer yang memberikan metodologi untuk penalaran tentang informasi yang ada dalam basis pengetahuan dan dalam *workplace* (Listiyono, 2008).

5. Fasilitas penjelasan (*explanation facilities)*

Fasilitas penjelas adalah komponen tambahan yang akan meningkatkan kemampuan sistem pakar. Komponen ini menggambarkan penalaran sistem kepada pemakai (Listiyono, 2008).

6. Perbaikan pengetahuan

Pakar memiliki kemampuan untuk menganalisa dan meningkatkan kinerja serta kemampuan untuk belajar dan kinerjanya. Kemampuan tersebut adalah penting dalam pembelajaran komputerisasi, sehingga program akan mampu menganalisis penyebab kesuksesan dan kegagalan yang dialaminya (Listiyono, 2008).

2.3.2 Penyakit Anak Pra Setahun

1. Diare

Diare adalah penyakit yang membuat penderitanya menjadi sering buang air besar, dengan kondisi tinja yang encer. Pada umumnya diare terjadi akibat makanan dan minuman yang terpapar virus, bakteri, atau parasit (Willy, 2017).

2. Dehidrasi

Dehidrasi adalah kondisi ketika tubuh kehilangan lebih banyak cairan daripada yang didapatkan, sehingga keseimbangan zat gula dan garam menjadi terganggu, akibatnya tubuh tidak dapat berfungsi secara normal (Marianti, 2018).

3. Demam

Demam adalah kondisi ketika suhu tubuh berada di atas angka 38 derajat celcius. Demam merupakan bagian dari proses kekebalan tubuh yang sedang melawan infeksi akibat virus, bakteri, atau parasit. Selain itu demam juga bisa terjadi pada kondisi hipertiroidisme, artritis, atau karena penggunaan beberapa jenis

obat-obatan, termasuk antibiotik. Kenaikan suhu tubuh akibat konsumsi obat ini disebut dengan demam obat (*drug fever*) (Marianti, 2017).

2.3.3 *Forward Chaining*

Forward Chaining adalah teknik pencarian yang dimulai dengan fakta yang diketahui, kemudian mencocokkan fakta-fakta tersebut dengan bagian *IF* dari *rules IF-THEN*. Bila ada fakta yang cocok dengan bagian *IF*, maka *rule* tersebut dieksekusi. Bila sebuah *rule* dieksekusi, maka sebuah fakta baru (bagian *THEN*) ditambahkan ke dalam *database*. Setiap kali pencocokan, dimulai dari *rul*e teratas. Setiap *rule* hanya boleh dieksekusi sekali saja. Proses pencocokan berhenti bila tidak ada lagi *rule* yang bisa dieksekusi. Pendekatan dalam pelacakan dimulai dari informasi masukan dan selanjutnya mencoba menggambarkan kesimpulan, pelacakan kedepan mencari fakta yang sesuai dengan bagian *IF* dari aturan *IF-THEN*. Dengan metode *forward chaining* dari pendekatan dan aturan yang telah dihasilkan dapat ditinjau oleh para ahli untuk diperbaiki atau dimodifikasi untuk memperoleh hasil yang lebih baik (Verina, 2015).

Untuk mempermudah pemahaman mengenai metode ini, akan diberikan ilustrasi kasus pembuatan sistem pakar dengan daftar aturannya sebagai berikut:

R1: Jika Premis 1 Dan Premis 2 Dan Premis 3 Maka Konklusi 1

R2: Jika Premis 1 Dan Premis 3 Dan Premis 4 Maka Konklusi 2

R3: Jika Premis 2 Dan Premis 3 Dan Premis 5 Maka Konklusi 3

R4: Jika Premis 1 Dan Premis 4 Dan Premis 5 Dan Premis 6 Maka Konklusi 4

Penelusuran maju pada kasus ini adalah untuk mengetahui apakah suatu fakta yang dialami oleh pengguna itu termasuk konklusi 1, konklusi 2, konklusi 3, atau konklusi 4 atau bahkan bukan salah satu dari konklusi tersebut, yang artinya sistem belum mampu mengambil kesimpulan karena terbatas aturan. Seandainya user memilih premis 1, premis 2, dan premis 3, maka aturan yang terpilih adalah aturan R1 dengan konklusinya adalah konklusi 1. Seandainya *user* memilih premis 1 dan premis 6, maka sistem akan mengarah pada aturan R4 dengan konklusinya adalah konklusi 4, tetapi karena aturan tersebut premisnya adalah premis 1, premis

4, premis 5, dan premis 6, maka premis-premis yang dipilih oleh user tidak cukup untuk mengambil kesimpulan konklusi 4 sebagai konklusi terpilih.

2.3.4 Analisis Sistem

Kegiatan analisis sistem adalah kegiatan untuk melihat sistem yang sudah berjalan, melihat bagian mana yang baik dan yang perlu dikembangkan atau di perbaiki, dan kemudian mendokumentasikan kebutuhan yang akan dipenuhi dalam sistem yang baru. Hal tersebut terlihat sederhana, namun sebenarnya tidak. Banyak hambatan yang akan ditemui dalam proses tersebut (Rosa dan Shalahuddin, 2018).

2.3.5 Aplikasi *Web*

Secara istilah pengertian aplikasi adalah suatu program yang siap untuk digunakan yang dibuat untuk melaksanakan suatu fungsi untuk pengguna yang dituju. *Web* adalah salah satu aplikasi yang berisikan dokumen-dokumen multimedia (teks, gambar, suara, animasi, video) didalamnya yang menggunakan protokol HTTP (*hyper text transfer protocol*) dan untuk mengaksesnya menggunakan perangkat lunak yang disebut *browser* (Irawan, dkk 2018).

2.3.6 *Codeigniter*

CodeIgniter merupakan salah satu *framework* PHP yang popular. *CodeIgniter* tergolong *framework* dengan ukuran kecil dan cukup mudah dikuasai. *codeigniter* juga datang dengan manual yang tergolong lengkap. *Codeigniter* merupakan aplikasi sumber terbuka yang berupa *framework* PHP dengan model MVC (*Model, View, Controller*) untuk membangun *website* dinamis dengan menggunakan PHP. *CodeIgniter* memudahkan *developer* untuk membuat aplikasi *web* dengan cepat mudah dibandingkan dengan membuatnya dari awal.

Fungsi penggunaan *CodeIgniter* pertama adalah *CodeIgniter* akan menghasilkan suatu struktur pemrograman yang sangat rapi, baik dari segi kode maupun struktur file PHPnya dikarenakan *CodeIgniter* dibangun berbasis MVC (*Model, View, Controller*) yang memisahkan antara tampilan dan logika aplikasi.

1. *Model* adalah bagian yang bertanggungj awab terhadap operasi *database*, baik itu *create*, *read*, *update*, dan *delete*. *Model* berupa fungsi-fungsi operasional *database* yang dapat dipanggilkan oleh *controller*.
2. *View* adalah bagian menangani tampilan, bagian inilah yang bertugas untuk mempresentasikan data kepada *user*. *View* berbentuk struktur HTML yang berisikan variabel data yang dikirimkan oleh *Controller*.
3. *Controller* adalah bagian yang mengatur hubungan antar *Model* dan *View*. *Controller* adalah otak dari kinerja aplikasi. *Controller* terdiri dari fungsi-fungsi yang bersifat operasional dan logikal (Nur, 2017).

2.3.7 *Bootstrap*

Bootstrap merupakan sebuah alat bantu untuk membuat sebuah tampilan halaman *website* yang dapat mempercepat pekerjaan seorang pengembang *website* ataupun pendesain halaman *website*. Sesuai namanya, *website* yang dibuat oleh alat bantu ini memiliki tampilan halaman yang sama atau mirip dengan tampilan halaman twitter atau desainer juga dapat mengubah tampilan halaman *website* sesuai dengan kebutuhan.Tampilan *website* yang dibuat *bootstrap* akan menyesuaikan ukuran layar dari *browser* yang kita gunakan baik *dekstop*, tablet ataupun *mobile device*. Fitur ini bisa diaktifkan ataupun di-non-aktifkan sesuai keinginan. Dengan *bootstrap* kita juga bisa membangun *web* dinamis ataupun statis (Wahyu, 2014).

2.3.8 UML

Unified Modeling Language (UML) adalah sebuah bahasa yang berdasarkan grafik atau gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan *software* berbasis OO (*object-oriented*) (Suendri, 2018).

2.3.9 *Object Oriented Programming*

OOP (*object oriented programming)* adalah sebuah istilah yang diberikan kepada bahasa pemrograman yang menggunakan tehnik berorientasi atau berbasis pada sebuah obyek dalam pembangunan program aplikasi, maksudnya bahwa orientasi pembuatan program tidak lagi menggunakan orientasi linear melainkan berorientasi pada objek-objek yang terpisah-pisah. Suatu perintah dalam bahasa ini diwakili oleh obyek yang didalamnya berisi beberapa perintah-perintah standar sederhana (Danuri, 2009).

Pemrograman berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi. Pemrograman berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Pemrograman berorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek (Rosa dan Shalahuddin, 2018).

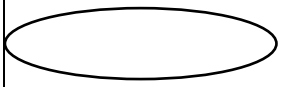
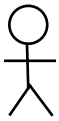
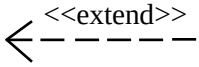
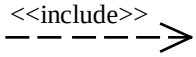
2.3.10 *Use Case Diagram*

Use case diagram digunakan untuk menggambarkan sistem dari sudut pandang pengguna sistem tersebut. Sehingga pembuatan *use case* diagram lebih dititik beratkan pada fungsionalitas yang ada pada sistem, bukan berdasarkan alur atau urutan kejadian. Sebuah *use case diagram* mempresentasikan sebuah interaksi antara aktor dengan sistem (Isa dan Hartawan, 2017).

Use case diagram merupakan pemodelan untuk melakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut.

Simbol-simbol yang digunakan dalam *Use Case Diagram* ditunjukan pada Tabel 2.1.

Tabel 2.1 Simbol-simbol U*se Case*.

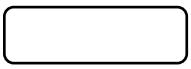
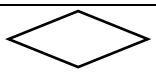
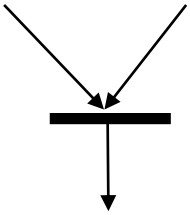
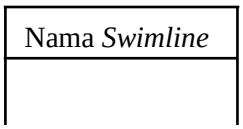
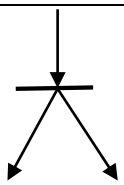
NO	SIMBOL	NAMA	KETERANGAN
1.		*Use case*	Menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, yang dinyatakan dengan menggunakan kata kerja
2.		*Actor*	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari 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 kontrol terhadap *use case*
3.		*Association*	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengindikasikan data.
4.		*Extend*	Merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.
5.		*Generalization*	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengindikasikan bila aktor berinteraksi secara pasif dengan sistem.
6.		*Include*	Merupakan didalam *use case* lain (*required*) atau pemanggilan *use case oleh use case* lain, contohnya adalah pemanggilan sebuah fungsi program

2.3.11 *Activity Diagram*

Menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktivitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktivitas lainnya. Diagram ini sangat mirip dengan *flowchart* karena memodelkan *workflow* dari suatu aktivitas ke status. Pembuatan *activity* diagram pada awal pemodelan proses dapat membantu memahami keseluruhan proses. *Activity diagram* juga digunakan untuk menggambarkan interaksi antara beberapa *use case* (Isa dan Hartawan, 2017).

Activity diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram* ditunjukan pada Tabel 2.2.

Tabel 2.2 Simbol-simbol A*ctivity Diagram.*

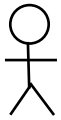
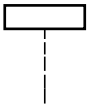
NO	GAMBAR	NAMA	KETERANGAN
1		*Start*	*Start point,* diletakan pada pojok kiri atas dan merupakan awal aktivitas
2		*Activities*	Menggambarkan suatu proses kegiatan.
3		*Decision*	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4		*Join*	*Join* (penggabungan) atau *rake,* digunakan untuk menunjukan adanya dekomposisi
5.		*Final*	Status akhir yang dilakukan sistem.
6.		*Swimlane*	Pembagian *activity diagram* untuk menunjukan siapa melakukan apa
7.		*Fork*	Percabangan digunakan untuk menunjukan kegiatan yang dilakukan secara paralel

2.3.12 *Sequence Diagram*

Menggambarkan interaksi antara sejumlah objek dalam urutan waktu. Kegunaannya untuk menunjukan rangkaian pesan yang dikirim antar objek juga interaksi antar objek yang terjadi pada titik tertentu dalam eksekusi sistem (Isa dan Hartawan, 2017).

Sequence diagram menggambarkan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram* ditunjukan pada Tabel 2.3.

Tabel 2.3 Simbol-simbol *Sequence Diagram.*

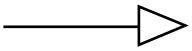
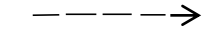
NO.	SIMBOL	NAMA	KETERANGAN
1.		*Actor*	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem
2.		*Self message*	Menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri
3.		*Message*	Simbol mengirim pesan antar *class*
4.		*Activation*	Mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivasi sebuah operasi
5.		*Lifeline*	Garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*

2.3.13 *Class Diagram*

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem.

Class Diagram juga menunjukan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class Diagram* secara khas meliputi kelas (*Class), relation assosiations, generalitation* dan *aggregation, attributes, method*, dan *visibility* tingkat akses objek eksternal kepada suatu operasi atau attribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau *cardinality* (Hendini, 2016).

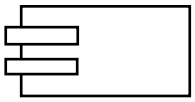
Tabel 2.4 Simbol-simbol *Class Diagram*

NO	SIMBOL	NAMA	KETERANGAN
1.	Nama kelas + Atribut + Operasi()	*Class*	Kelas pada struktur sistem
2.		*Association*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity.*
3.		*Directed association*	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain biasanya juga disertai dengan *multiplicity.*
4.		*Generalization*	Relasi antar kelas dengan makna generalisasi - spesialisai (umum – khusus).
5.		*Dependency*	Relasi antar kelas dengan makna kebergantungan antar kelas.

2.3.14 *Component Diagram*

Component diagram menggambarkan struktur fisik dari kode, pemetaan pandangan logis dari kelas proyek untuk kode aktual di mana logika ini di laksanakan Irawan, dkk. (2018)

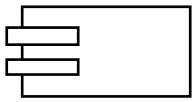
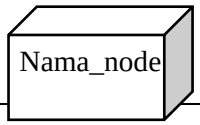
Tabel 2.5 Simbol-simbol *Component Diagram*

NO	SIMBOL	NAMA	KETERANGAN
1.		*Component*	Pada *component diagram*, komponen-komponen yang ada diletakan didalam *node* untuk memastikan keberadaan posisi mereka
2.		*Dependency*	Simbol yang menjelaskan sebuah keterkaitan antara komponen, satu komponen dengan yang lain. Arah panah dalam simbol tersebut diarahkan pada komponen yang dipakai.
3.		*Communication path*	Simbol ini dipakai untuk mengarahkan relasi antar komponen, jika suatu komponen memiliki relasi atau keterkaitan dengan komponen lainnya maka dipakailah simbol link ini.

2.3.15 *Deployment Diagram*

Deployment Diagram digunakan untuk menggambarkan detail bagaimana komponen disusun di infrastruktur sistem. Simbol-simbol *deployment diagram* ditunjukan pada Tabel 2.6.

Tabel 2.6 Simbol-simbol *Deployment Diagram*.

NO	SIMBOL	NAMA	KETERANGAN
1.		*Component*	Pada *deployment diagram,* komponen-komponen yang ada diletakan didalam *node* untuk memastikan keberadaan posisi mereka
2.		*Node*	*Node* menggambarkan bagian *hardware* dalam sebuah sistem. Notasi untuk *node* digambarkan sebagai sebuah kubus tiga dimensi
3.		*Association / Comumunication Path*	Menghubungkan dua node yang mengindikasikan jalur komunikasi antara elemen-elemen *hardware.*

2.3.16 Metode Pengembangan Sistem

Metode pengembangan sistem yang diterapkan pada penelitian ini adalah dengan pengembangan metode *waterfall*. Metode *waterfall* merupakan model pengembangan sistem informasi yang sistematik dan sekuensial (Sasmito, 2017). Metode *waterfall* memiliki tahapan-tahapan sebagai berikut :

1. *Requirements analysis and definition*

Layanan sistem, kendala, dan tujuan ditetapkan oleh hasil konsultasi dengan pengguna yang kemudian didefinisikan secara rinci dan berfungsi sebagai spesifikasi sistem.

2. *System and software design*

Tahapan perancangan sistem mengalokasikan kebutuhan-kebutuhan sistem baik perangkat keras maupun perangkat lunak dengan membentuk arsitekur sistem secara keseluruhan. Perancangan perangkat lunak melibatkan identifikasi dan penggambaran abstraksi sistem dasar perangkat lunak dan hubunganya.

3. *Implementation and unit testing*

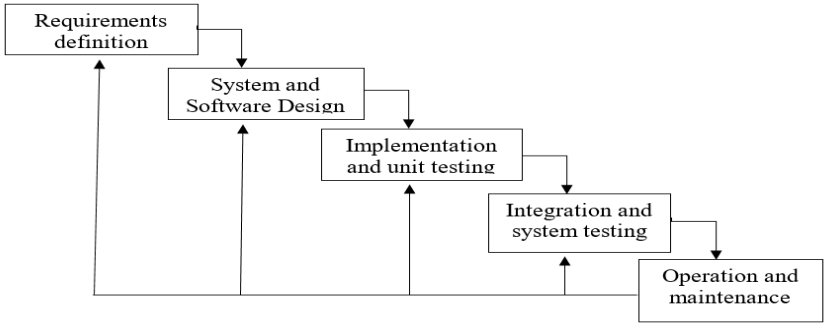
Pada tahap ini, perancangan perangkat lunak direalisasikan sebagai serangkaian program atau unit program. Pengujian melibatkan verifikasi bahwa setiap unit memenuhi spesifikasinya.

4. *Integration and system testing*

Unit-unit individu program atau program digabung dan diuji sebagai sebuah sistem lengkap dengan untuk memastikan apakah sesuai dengan kebutuhan perangkat lunak atau tidak. Setelah pengujian perangkat lunak dapat dikirimkan ke *customer*.

5. *Operation and maintenance*

Sistem dipasang dan digunakan secara nyata. *Maintenance* melibatkan pembetulan kesalahan yang tidak ditemukan pada tahapanya melibatkan pembetulan kesalahan yang tidak ditemukan pada tahapan-tahapan sebelumnya, meningkatkan implementasi dari unit sistem, dan meningkatkan layanan sistem sebagai kebutuhan baru. Bagan metode *waterfall* ditunjukan pada Gambar 2.2.



Gambar 2.2 Metode *Waterfall*

2.3.17 Metode Pengujian *Webqual*

Metode *webqual* merupakan salah satu metode atau teknik pengukuran kualitas *website* berdasarkan persepsi pengguna. *Webqual* sudah mulai dikembangkan sejak tahun 1998 dan mengalami beberapa interaksi dalam penyusunan dimensi dan butir pertanyaannya. Dimensi dari *website* yang diinginkan oleh pengguna ada tiga yaitu dilihat dari dimensi kemudahan penggunaan (*Usability Quality*), kualitas informasi (*Information Quality*) dan

kualitas interaksi layanan (*Service Interaction Quality*) (Nugraha dan Silfianti, 2016).

Model pengukuran mutu *website* dengan kuesionernya berdasarkan 3 dimensi *webqual* dapat dilihat pada Tabel 2.7.

Tabel 2.7 *Webqual* 4.0 instrumen.

Dimensi	Kuesioner *Webqual* 4.0
Usability	*I find the site easy to learn to operate*
	My interaction with the site is clear and understandable
	I find the site easy to navigate
	I find the site easy to use
	The site has an attractive appearance
	The design is appropriate to the type of site
	The site conveys a sense of competency
	The site creates a positive experience
Information Quality	*Provides accurate information*
	Provides believable information
	Provides timely information
	Provides relevant information
	Provides easy to understand information
	Provides information at the right level of detail
	Present the information in appropriate format
Service Interaction	*Has a good reputation*
	It feels safe to complete transaction
	My personal information feels secure
	Creates a sense of personalization
	Convey a sense of community
	Make it easy to communicate with the organization
	I feel confident that goods/services will be delivered as promised
Overall	*Overall View of The Website*

2.3.18 Skala likert

Skala likert adalah skala yang digunakan untuk mengukur persepsi, sikap atau pendapat seseorang atau kelompok mengenai sebuah peristiwa atau fenomena sosial. Terdapat dua bentuk pertanyaan dalam skala likert, yaitu bentuk pertanyaan positif untuk mengukur skala positif, dan bentuk pertanyaan negatif untuk mengukur skala negatif. Pertanyaan positif diberi skor 5, 4, 3, 2, dan 1. Sedangkan bentuk pertanyaan negatif diberi skor 1, 2, 3, 4 dan 5 (Pranatawijaya, dkk., 2019)