BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka

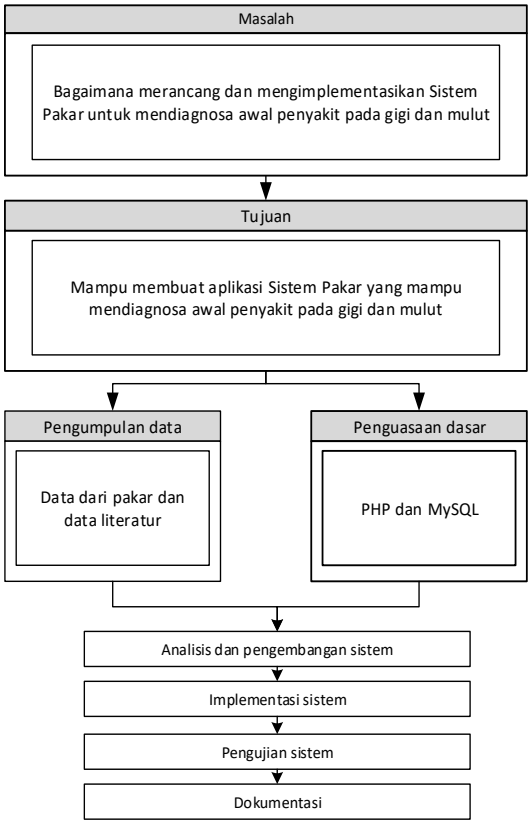
Arfajsyah, H. S., Permana, I., & Salisah, F. N. (2018) mengimplementasikan Sistem Pakarnya yang berjudul "Sistem pakar berbasis *Android* untuk diagnosa penyakit gigi dan mulut". Pada penelitian ini, aplikasi sistem pakar dibuat berbasis *mobile* dengan *platform* yang digunakan adalah *Android*. Aplikasi dibuat berbasis *mobile* agar aplikasi yang dibuat bisa digunakan kapan saja dan dimana saja. Metode inferensi yang digunakan pada penelitian ini adalah *forward chaining*. Metode inferensi adalah teknik inferensi yang didasari dengan fakta-fakta yang diketahui, kemudian mencocokan. Berdasarkan hasil uji *black box, unit test*, dan *UAT* dapat disimpulkan bahwa aplikasi ini bisa direkomendasikan untuk digunakan oleh masyarakat sehingga dapat membantu masyarakat melakukan diagnosa awal penyakit gigi dan mulut.

Napianto, R., Rahmanto, Y., & Lestari, R. I. B. D. O. (2019). Dalam jurnal yang berjudul "*Software Development* Sistem Pakar Penyakit Kanker Pada Rongga Mulut Berbasis *Web*". Pada pengembangan sistem pakar penyakit kanker pada rongga mulut menggunakan pendekatan metode pengembangan sistem *extreme programming* (*XP*). *XP adalah software development* yang memiliki sasaran untuk *software* skala kecil sampai medium. Penelitian ini menghasilkan aplikasi sistem pakar diagnosa penyakit kanker rongga mulut yang berhasil mendiagnosa penyakit dan solusi berdasarkan *rules based* yang telah dibangun sebelumnya.

Yansyah, I. R., & Sumijan, S. (2021). Dalam jurnal yang berjudul "Sistem Pakar Metode *Forward Chaining* untuk Mengukur Keparahan Penyakit Gigi dan Mulut" dijelaskan panjangnya antrian yang menyebabkan pasien kurang betah dalam menunggu. Untuk mengatasi permasalahan tersebut dilakukanlah penelitian sehingga diperlukan sebuah aplikasi sistem pakar untuk melakukan diagnosa sejak dini tingkat keparahan penyakit gigi dan mulut yang diderita sehingga nantinya didapatkan solusi penanganannya sebelum dibawa ke dokter.

2.2 Kerangka Pemikiran

Kerangka Pemikiran adalah penjelasan sementara secara konseptual tentang keterkaitan hubungan pada setiap objek permasalahan berdasarkan teori. Kerangka pemikiran Sistem Pakar untuk mendiagnosa penyakit gigi dan mulut di Rumah Sakit Gigi dan Mulut Universitas Airlangga Surabaya ditunjukan pada Gambar 2.



Gambar 2.1 Diagram Kerangka Pemikiran.enelitian

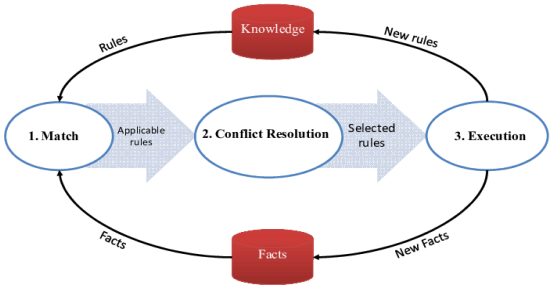
2.3 Teori Pendukung

2.3.1 Sistem Pakar

N. Jarti and R. Trisno. (2017). Sistem Pakar dikatakan sistem mengadopsikan cara kerja atau pengetahuan manusia ke komputer yang dirancang untuk memodelkan kemampuan masalah seperti seorang pakar. Sistem pakar adalah sistem berbasis komputer yang menggunakan pengetahuan, fakta dan teknik penalaran dalam memecahkan masalah yang biasanya hanya dapat dipecahkan oleh seorang pakar dalam bidang tersebut.

2.3.2 Sistem berbasis aturan atau *rules*

Dahria, M. (2011). Sistem pakar yang dibuat merupakan sistem yang berdasarkan pada aturan–aturan dimana *program* disimpan dalam bentuk aturan-aturan sebagai prosedur pemecahan masalah. Aturan tersebut biasanya berbentuk *IF–THEN* terlebih dahulu. *Rules* adalah sebuah struktur *knowledge* yang menghubungkan beberapa informasi yang sudah diketahui ke informasi lain sehingga dapat disimpulkan. Sebuah *rules* adalah sebuah bentuk *knowledge* yang *procedural*. Dengan demikian yang dimaksud dengan sistem pakar berbasis *rules* adalah sebuah program komputer untuk memproses masalah dari informasi spesifik yang terdapat dalam memori aktif dengan sebuah *set* dari *rules* dalam *knowledge base*, dengan menggunakan *inference engine* untuk menghasilkan informasi baru. Atas dasar tersebut dapat dilihat gambar 2.2



Gambar 2.2 Proses berbasis aturan.

2.3.3 PHP

Haryana, K. S. (2015). PHP merupakan salah satu bahasa pemograman berbasis *web* dimana sistem yang diterapkan adalah pada sisi *server side*. PHP dapat disisipkan diantara skrip-skrip bahasa HTML dan arena bahasa *server side* lainnya, dengan itu maka PHP akan dieksekusi secara langsung pada *server*. Sedangkan *browser* akan mengeksekusi halaman *web* tersebut melalui *server* yang kemudian akan menerima tampilan “hasil jadi” dalam bentuk HTML, sedangkan kode PHP itu sendiri tidak akan dapat terlihat.

2.3.4 MySQL

Enterprise, J. (2018). MySQL merupakan *server* yang melayani *database*. Untuk membuat dan mengolah *database*, kita dapat mempelajari pemrograman khusus yang disebut *query* (perintah) SQL. *Database* sendiri dibutuhkan jika kita

ingin meng*input* data dari user menggunakan *form* HTML, untuk kemudian diolah PHP agar bisa disimpan ke dalam *database* MySQL.

2.3.5 Web

Sugono, D., (2008) Menurut kamus Bahasa Indonesia *web* berarti sistem untuk mengakses, memanipulasi, dan mengunduh dokumen hiperatut yang terdapat dalam komputer yang dihubungkan melalui internet. Sebuah halaman *web* biasanya berupa dokumen yang ditulis dalam format HTML, yaitu sebuah *protocol* yang menyampaikan informasi dari *server website* untuk ditampilkan kepada para pemakai melalui *web browser*. Semua publikasi dari *website* tersebut dapat membentuk sebuah jaringan informasi yang sangat besar.

2.3.6 Diagnosa

Sugono, D., (2008) Diagnosa atau diagnosis dalam kamus besar Bahasa Indonesia adalah penentuan suatu penyakit dengan meneliti (memeriksa) gejala-gejalanya.

2.3.7 Penyakit Gigi dan Mulut

Makarios, A., & Prasetiyowati, M. I. (2012) Penyakit adalah suatu proses penghancuran secara partikular dalam suatu organ atau organisme yang menyebabkan keadaan tidak nyaman yang dialami tubuh dan pikiran. Sedangkan untuk pengertian penyakit gigi dan mulut adalah proses penghancuran secara partikular yang terjadi pada gigi dan mulut. Walaupun amat jarang terjadi, bahaya yang datang dari penyakit gigi dan mulut terkadang bisa sangat berbahaya.

2.3.8 *Database*

Latief, M. (2012). *Database* merupakan kumpulan dari item data yang saling berhubungan satu dengan yang lainnya yang diorganisasikan berdasarkan sebuah skema atau struktur tertentu, tersimpan di *hardware* komputer dan dengan *software* untuk melakukan manipulasi untuk kegunaan tertentu. *Database* dapat juga diartikan, koleksi data yang terorganisasi untuk melayani beragam aplikasi secara efisien dengan mensentralisasi data dan meminimalisasi data yang berlebih.

2.3.9 *Codeigniter*

Somya, R. (2018). *CodeIgniter* adalah sebuah *framework* PHP yang berupa kumpulan folder dan file *PHP*, *JavaScript*, *CSS*, *TXT*, dan file berbasis WEB

lainnya dengan setting tertentu untuk menggunakannya dan menyediakan *library* dan *helper* yang dapat dimanfaatkan di dalam pemrograman PHP. *CodeIgniter* tergolong *framework* dengan ukuran kecil dan cukup mudah dikuasai, *CodeIgniter* membutuhkan web *server* agar dapat dijalankan.

Codeigniter memiliki kelebihan jika dibandingkan dengan framework lain adalah sebagai berikut:

a. Gratis (*Open-Source*) Kerangka kerja *Codeigniter* memiliki lisensi dibawah *Apache/BSD open-source* sehingga bersifat bebas atau gratis.
b. Berukuran kecil Ukuran yang kecil merupakan keunggulan tersendiri jika dibandingkan *framework* lain yang berukuran besar dan membutuhkan *resource* yang besar dan juga dalam eksekusi maupun penyimpanannya.
c. Menggunakan konsep M-V-C *Codeigniter* merupakan konsep M-V-C (*ModelView-Controller*) yang memungkinkan pemisahan antara *layer application-logic dan presentation*. Dengan konsep ini kode *PHP, query Mysql, Javaascript* dan CSS dapat saling dipisah-pisahkan sehingga ukuran file menjadi lebih kecil dan lebih mudah dalam perbaikan kedepannya atau *maintenance*.
    a) Model Kode merupakan program (berupa OOP class) yang digunakan untuk berhubungan dengan database *MySQL* sekaligus untuk memanipulasinya (*input-edit-delete*).
    b) *View* Merupakan kode program berupa template atau PHP untuk menampilkan data pada *browser*.

    Destiningrum, M., & Adrian, Q. J. (2017). *Controller* merupakan Kode program (berupa OOP *class*) yang digunakan untuk mengontrol aliran atau dengan kata lain sebagai pengontrol *model* dan *view* .

2.3.10 *Bootstrap*

Somya, R. (2018). *Bootstrap* merupakan salah satu *framework* CSS yang sering digunakan untuk memperindah tampilan suatu *website*. Tujuan dari *Boostrap* adalah mempercepat pekerjaan. *Framework* ini sering digunakan oleh *front-end* programmer namun tidak menutup kemungkinan juga apabila digunakan oleh *back-end programmer*. Kelebihan dari *Bootstrap* ini adalah tidak hanya membuat

tampilan yang statis namun dapat membuat tampilan dinamis dan beberapa animasi dengan bantuan *plugin JavaScript*. Selain itu juga, *Bootstrap* mendukung untuk membuat *web* responsif, yaitu tampilan akan berubah ukurannya tergantung pada resolusi layar *device* yang digunakan oleh *user* .

2.3.11 UML *(Unified Modelling Language)*

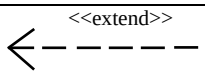
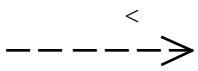
Effendy, & Mulyono, H. (2020). UML merupakan singkatan dari "*Unified Modelling Language*" yaitu suatu metode permodelan secara visual untuk sarana perancangan sistem berorientasi objek, atau definisi UML yaitu sebagai suatu bahasa yang sudah menjadi standar pada visualisasi, perancangan dan juga pendokumentasian sistem *software*. Saat ini UML sudah menjadi bahasa standar dalam penulisan *blue print software.*

UML adalah salah satu alat bantu yang dapat digunakan dalam bahasa pemrograman yang berorientasi objek.

2.3.12 *Use case Diagram*

Maiyedra, N. A. (2019). *Use case diagram* adalah deskripsi fungsi dari sebuah sistem dari perspektif penggguna. *Use case diagram* bekerja dengan cara mendeskripsikan tipikal interaksi antara user sebuah sistem dengan sistemnya sendiri melaluhi sebuah cerita bagaimana sebuah sistem dipakai . Elemen-elemen yang digunakan dalam pembuatan *use-case diagram* umumnya terdiri dari *use-case, actor, Association, Extend, Generalization* dan *Include* seperti yang ditunjukkan pada Tabel 2.1.

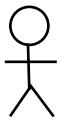
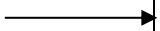
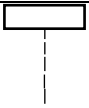
Tabel 2.1 Elemen-Elemen *Use-Case Diagram*

NO	SIMBOL	NAMA	KETERANGAN
1.		*Use case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
2.		*Actor*	Aktor adalah himpunan peran yang pengguna mainkan ketika berinteraksi dengan *usecase*.
3.		*Association*	Menghubungkan antara objek satu dengan objek yang lain.
4.		*Extend*	Menspesifikasikan bahwa *usecase* target memperluas perilaku dari *usecase* sumber pada suatu titik yang diberikan.
5.		*Generalization*	Hubungan dimana objek anak(*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk(*ancestor*).
6.		*Include*	Menspesifikasikan bahwa *usecase* sumber secara eksplisit.

2.3.13 *Sequance Diagram*

Maiyedra, N. A. (2019). *Sequence diagram* menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. Tabel 2.2 adalah elemen-elemen yang digunakan dalam pembuatan *sequence diagram.*

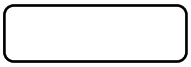
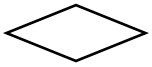
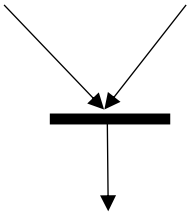
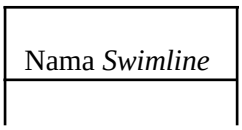
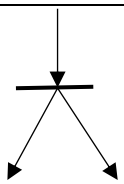
Tabel 2.2 Elemen-Elemen *Sequance Diagram*

NO.	SIMBOL	NAMA	KETERANGAN
1.		*Actor*	Aktor adalah himpunan peran yang pengguna mainkan ketika berinteraksi dengan *sequance diagram*.
2.		*Self message*	Menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri
3.		*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi tentang aktifitas yang terjadi.
4.		*Activation*	Mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivasi sebuah operasi
5.		*Lifeline*	Objek *entity* antarmuka yang saling berinteraksi.

2.3.14 *Activity Diagram*

Maiyedra, N. A. (2019). *Activity diagram* adalah teknik untuk mendiskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. Tabel 2.3 adalah elemen-elemen yang digunakan dalam pembuatan *activity diagram.*

Tabel 2.3 Elemen-Elemen *Activity Diagram*

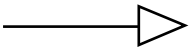
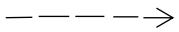
NO	GAMBAR	NAMA	KETERANGAN
1		*Start*	Bagaimana objek dibentuk atau diawali.
2		*Activities*	*State* dari sistem yang mencerminkan eksekusi dari suatu aksi.
3		*Decision*	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4		*Join*	*Join* (penggabungan) atau *rake*, digunakan untuk menunjukan adanya dekomposisi
5.		*Final*	Bagaimana objek dibentuk dan dihancurkan.
6.		*Swimlane*	Pembagian *activity diagram* untuk menunjukan siapa melakukan apa
7.		*Fork*	Percabangan digunakan untuk menunjukan kegiatan yang dilakukan secara paralel

2.3.15 *Class Diagram*

Maiyedra, N. A. (2019*). Class diagram* adalah deskripsi kelompok obyek-obyek dengan properti, perilaku dan relasi yang sama. Sehingga dengan adanya *class diagram* dapat memberikan pandangan global atas sebuah sistem. Hal tersebut tercermin dari *class-class* yang ada dan relasinya satu dengan yang lainnya. Sebuah

sistem biasanya mempunyai beberapa *class diagram*. Tabel 2.4 adalah elemen-elemen yang digunakan dalam pembuatan *class diagram.*

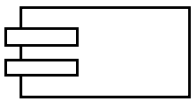
Tabel 2.4 Elemen-Elemen *Class Diagram*

NO	SIMBOL	NAMA	KETERANGAN
1.	Nama kelas + Atribut + Operasi()	*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
2.		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
3.		*Directed association*	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain biasanya juga disertai dengan *multiplicity*.
4.		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
5.		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri

2.3.16 *Component Diagram*

Kusuma, A. M., & Yosrita, E. (2016). *Component diagram* menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya. Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik library maupun *executable*, baik yang muncul pada *compile time, link time*, maupun *run.* Tabel 2.5 adalah elemen-elemen yang digunakan dalam pembuatan *component diagram.*

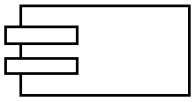
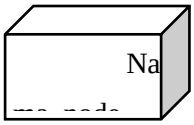
Tabel 2.5 Elemen-Elemen *Component Diagram*

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.

2.3.17 Deployment Diagram

Maiyedra, N. A. (2019). *Deployment diagram* menunjukkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian *software* yang berjalan pada bagian *hardware*. Tabel 2.6 adalah elemen-elemen yang digunakan dalam pembuatan *deployment diagram.*

Tabel 2.6 Elemen-Elemen 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.*

2.3.18 Pengujian *Blackbox*

Salamah, U. (2017), Pengujian *blackbox* merupakan salah satu jenis metode pengujian yang memperlakukan perangkat lunak yang tidak diketahui kinerja internalnya. Sehingga para tester memandang perangkat lunak seperti layaknya sebuah "kotak hitam" yang tidak penting dilihat isinya, tapi cukup dikenai proses testing dibagian luar. Pada jenis *blackbox testing*, perangkat lunak tersebut akan dieksekusi kemudian berusaha dites apakah telah memenuhi kebutuhan pengguna yang didefinisikan pada saat awal tanpa harus membongkar *listing* programnya. Pengujian dengan menggunakan *blackbox* mempunyai beberapa teknik, di antaranya *Equivalence Partitioning, Boundary Value Analysis, Robustness Testing, Behavior Testing, dan Cause-Effect Relationship Testing* (Safitri &Pramudita, 2018). Dalam pengujian *Boundary Value Analysis*, Teknik ini berfokus pada pencarian *error* dari luar atau sisi dalam perangkat lunak dilakukan melalui tahapan setiap modul dan menu yang ada pada Sistem pakar penyakit gigi dan mulut, pengujian dengan *Boundary Value Analysis* sehingga akan didapat nilai prosentase dari kegunaan sistem informasi ini.