BAB II
TINJAUAN PUSTAKA

II.1. Penelitian Terkait

Untuk mendukung keberhasilan penelitihan ini, penyusun melakukan pendekatan teoritis melalui beberapa literatur yang berhubungan dengan penelitihan yang dilakukan. Beberapa uraian penelitihan terdahulu yang menjadi acuan dalam penelitian ini yaitu:

Penelitian yang dilakukan Dwi Purnomo, dkk, 2017 yaitu Sistem "Pakar Diagnosa Penyakit Pada Kucing Menggunakan Metode *Dempster-Shafer* Berbasis Android" menghasilkan aplikasi aplikasi dilakukan dengan data rekam medis selama satu tahun terakhir, secara keseluruhan tingkat keberhasilan pengujian 94,59% dikarenakan adanya 2 penyakit yang memiliki gejala hampir sama dan mengharuskan pemeriksaan lebih lanjut ke ahli/pakar atau dokter hewan.

Penelitian dari Amalia Novi, dkk, 2019 yaitu Sistem Pakar Diagnosa Penyakit Kucing Dengan Metode *Dempster Shafer*Berbasis Web Penelitian Inimengahasilkan pengujian akurasi sebesar 88,88% dikarenakan adanya beberapa penyakit yang memiliki gejala hampir sama dan mengharuskan pemeriksaan lebih lanjut ke pakar atau dokter hewan.

Penelitian dari Nugroho,H,A, dkk, 2021 yaitu Sistem Pakar Diagnosa Penyakit Kucing Dengan Metode *Dempster Shafer*Berbasis Web Penelitian Ini

mengahasilkan dengan system pakar ini memudahkan seorang pakar untuk mendiagnosa penyakit pada kucing persia dengan cepat dan tepat sehingga mengurangai kesalahan dalam mendiagnosa, dikarenakan system ini menggunakan suatu metode yang cukup baik dalam dunia system pakar.

Penelitian dari Qori Istiqomah, dkk, 2020 yaitu Sistem Pakar Pendeteksi Penyakit Fus Pada Kucing Dengan Metode Dempster ShaferPenelitian Ini mengahasilkan Aplikasi sistem pakar ini berfungsi mendeteksi penyakit FUS menggunakan metode Dempster Shaferyang menarik kesimpulan penyakit FUS yang dialami dengan beberapa gejala yang ada dengan persentasenya. Hasil deteksi dan persentase besar kemungkinan diperoleh dengan menggunakan metode dempster shafer. Berdasarkan pengujian yang telah dilakukan, diperoleh persentase akurasi 96,1% sehingga dapat disimpulkan sistem dapat mendeteksi penyakit FUS dengan baik.

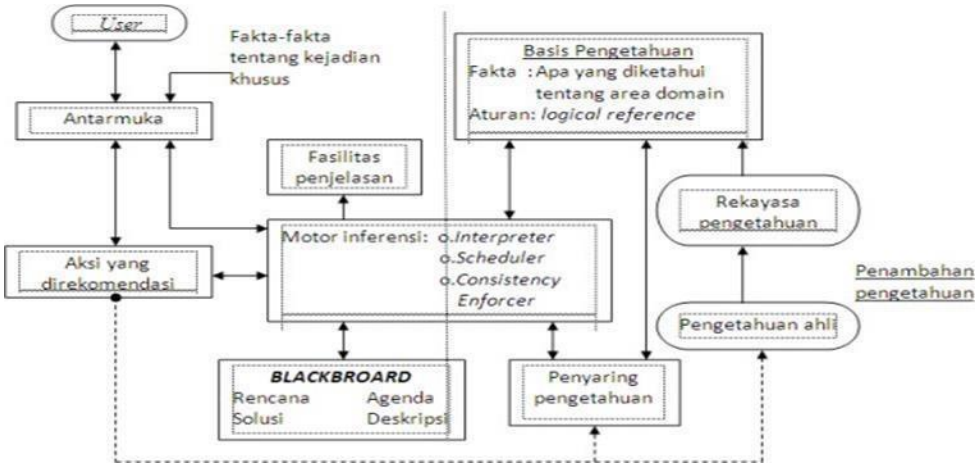
Penelitian dari Tataq Distasianto, dkk, 2020 yaitu Implementasi Metode Dempster-Shafer Untuk Diagnosa Penyakit Kulit Kucing Penelitian Ini Bahwa hasil yang direkomendasikan oleh sistem pakar telah cocok dan sesuai serta memiliki kesamaan sebesar 98,3% dengan hasil pakar.

II.2. Landasan Teoritis

II.2.1. Sistem Pakar

Sistem pakar adalah sistem yang menggunakan pengetahuan manusia yang

terekam dalam komputer untuk memecahkan persoalan yang biasanya memerlukan keahlian manusia. Sistem pakar yang baik dirancang agar dapat menyelesaikan suatu permasalahan tertentu dengan meniru kerja dari para ahli. Sistem pakar dapat ditampilkan dalam dua lingkungan, yaitu: pengembangan dan konsultasi. Lingkungan pengembangan digunakan oleh pembangun sistem pakar untuk membangun komponen dan memasukkan pengetahuan ke dalam basis pengetahuan. Lingkungan konsultasi digunakan oleh orang yang bukan ahli untuk memperoleh pengetahuan dan berkonsultasi. Komponen-komponen yang ada pada sistem pakar dapat dilihat pada Gambar 1, yaitu :



Gambar II.1 Sistem Pakar

II.2.2. Damster Shafer

Teori *Dempster-Shafer* merupakan teori matematika dari evidence. Teori tersebut dapat memberikan sebuah cara untuk menggabungkan evidence dari

beberapa sumber dan mendatangkan atau memberikan tingkat kepercayaan (direpresentasikan melalui fungsi kepercayaan) dimana mengambil dari seluruh evidence yang tersedia. Secara umum teori Dempster-Shafer ditulis dalam suatu interval : [Belief,Plausibility]. Belief (Bel) adalah ukuran kekuatan evidence dalam mendukung suatu himpunan proposisi. Jika bernilai 0 maka mengindikasikan bahwa tidak ada evidence, dan jika m bernilai 1 menunjukkan adanya kepastian. Plausibility juga bernilai 0 sampai 1. Jika kita yakin akan X', maka dapat dikatakan bahwa Bel(X')=1, sehingga rumus diatas nilai dari Pls(X')=0

Rumus Bel(x)= $\sum m(Y)$

Dan plausibility dinotasikan pada persamaan

$$\text{Pls(x)} = 1 - bel(X) = 1 - \sum m(X)$$

Dimana :

Bel(X) : Belief(X)

Pls(X) : Plausibility(X)

m(X) : mass function dari (X)

m(Y) : mass function dari (Y)

Perlu adanya probabilitas fungsi densitas atau mass function (m) yang merupakan tingkat kepercayaan dari suatu evidence (gejala). untuk mengatasi sejumlah evidence tersebut pada teori Dempster-Shafer menggunakan aturan yang lebih dikenal dengan Dempster's Rule of Combination :

$$m_3(Z) = \frac{\sum_{X \cap F = Z} m_1(X) m_2(Y)}{1 - \sum_{X \cap F = \emptyset} m_1(X) m_2(Y)}$$

Dengan :

m3(Z) = mass function dari evidence (Z)

m1(X) = mass function dari evidence (X)

m2(Y) = mass function dari evidence (Y) k = jumlah konflik evidence

Cara kerja metode Dempster-Shafer menurut :

1. Inisialisasi nilai densitas gejala berdasarkan pakar
2. Memilih gejala-gejala yang diinginkan
3. Mengambil nilai densitas (nilai belief) yang telah ditentukan pakar dari gejala
4. yang dipilih dan menghitung nilai plausibility menggunakan persamaan.
5. Menghitung nilai densitas (m) baru menggunakan tabel aturan kombinasi
6. Dari tabel kombinasi, didapatkan nilai m baru yang kemudian dihitung nilai probabilitas densitas menggunakan persamaan.
7. Kemudian, hitung nilai persentase derajat keyakinan dengan mengalikan 100% Pilihlah tingkat kepercayan yang paling besar

II.2.3. Abses

Abses adalah penumpukan nanah yang terjadi di dalam rongga bagian tubuh setelah terinfeksi. Pus yang terlokalisir akibat adanya infeksi dari supurasi jaringan merupakan abses yang bisa terjadi disemua jaringan atau struktur anatomi pertulangan. Kasus abses terjadi pada kucing dapat timbul akibat ada infeksi dari berbagai bakteri, salah satunya yaitu Pseudomonas aeruginosa (Gunawan et al., 2016;Hidayati, 2019).

Ada banyak potensi penyebab abses pada kucing. Salah satu penyebab paling umum adalah gigitan hewan lain. Luka tembus dari benda tajam seperti tongkat dan biji rumput juga dapat menyebabkan abses, seperti infeksi sebelumnya di lokasi tersebut (Weir dan Robin, 2021).

II.2.4. PHP (*Hypertext Preprocessor*)

Menurut Jubille Enterprise (2017:1) PHP adalah bahasa pemrograman yang digunakan untuk membuat sistem berbasis *website*. Sebagai sebuah sistem, *website* tersebut hendaknya memiliki sifat dinamis dan interaktif. Memiliki sifat dinamis, artinya *website* tersebut bisa berubah tampilan kontennya sesuai kondisi tertentu (misalnya, menampilkan produk yang berbeda-beda untuk setiap pengunjung). Artinya interaktif *website* tersebut dapat memberi *feedback* bagi *user* (misalnya, menampilkan hasil pencarian produk).PHP merupakan bahasa pemrograman berjenis *service-side*. Dengan demikian, PHP akan diproses oleh *server* yang hasil olahannya akan dikirim kembali ke *browser*. Oleh karena itu, salah satu *tool* yang harus tersedia sebelum memulai pemrograman PHP adalah *server*

II.2.5. Sublime Text

Sublime text merupakan perangkat lunak text editor yang digunakan untuk membuat atau meng-edit suatu aplikasi. Sublime text mempunyai fitur plugin tambahan yang memudahkan programmer (Supono dan Putratama, 2016:14).

II.2.6. Database

Database adalah sekumpulan tabel-tabel yang saling berelasi, relasi tersebut bisa ditunjukkan dengan kunci dari tiap tabel yang ada. Satu database menunjukkan satu kumpulan data yang dipakai dalam satu lingkup perusahaan atau instansi. Database mempunyai kegunaan dalam mengatasi penyusunan dan penyimpanan data, maka seringkali masalah yang dihadapi adalah:

1. Redudansi dan inkonsistensi data
2. Kesulitan dalam pengaksesan data
3. Isolasi data untuk standarisasi
4. Multiuser
5. Keamanan data
6. Integritas data
7. Kebebasan data (Urva dan Siregar, 2016:93)

II.2.7. MySQL

Menurut Solichin (2016:138), MySQL merupakan salah satu perangkat lunak basis data yang sangat populer. Saat ini tersedia versi MySQL yang berbayar (MySQL Enterprise Edition), namun tetap tersedia versi MySQL yang gratis (MySQL Community Edition). MySQL Community Edition dapat diunduh secara

gratis dan bebas digunakan dalam berbagai keperluan. Walaupun demikian, secara fitur dan kemampuan, MySQL versi tidak terlalu jauh berbeda dengan versi berbayar. Kelebihan MySQL adalah sebagai berikut:

1. Bersifat open source, yang memiliki kemampuan untuk dapat dikembangkan lagi. Menggunakan bahasa SQL (Structure Query Language), yang merupakan standar bahasa dunia dalam pengolahan data.

2. Super performance dan realible, tidak bisa diragukan, pemrosesan database-nya sangat cepat dan stabil.

3. Sangat mudah dipelajari (easy of use).

4. Memiliki dukungan support (group) penggunaan MySQL.

5. Multiuser, di mana MySQL dapat digunakan oleh beberapa user dalam waktu yang bersamaan tanpa mengalami konflik

II.2.8. UML *(Unified Modeling language)*

Unified Modeling Language (*UML*), adalah salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi objek. Hal ini disebabkan karena *UML* menyediakan bahasa pemodelan *visual* yang memungkinkan bagi pengembang sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk

berbagi *(sharing)* dan mengkomunikasikan rancangan mereka dengan yang lain.

UML merupakan keluarga notasi grafis yang didukung oleh meta-model tunggal yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek (OO). Pemodelan *visual* adalah salah satu cara berpikir tentang persoalan menggunakan model-model yang diorganisasikan seputar dunia nyata yang berguna untuk memahami persoalan, mengkomunikasikan dengan orang-orang yang terlibat dalam proyek (*costumer*, ahli dibidangnya, analisis, desainer dan lain-lain). Serta didefenisikan sebagai proses pemodelan sistem informasi menggunakan pengaturan standar elemen grafik dan objek-objek dalam sistem dan antar sistem itu sendiri. (Munawar; 2018: 49).

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

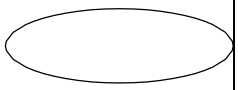
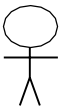
1. *Use case* Diagram

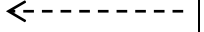
   Use Case Diagram merupakan pemodelan sistem informasi perilaku yangakan dilakukan. Kasus penggunaan menggambarkan interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan bahwause case memungkinkan untuk mengetahui apa saja fungsi dari sistem informasidan siapa yang berhak menggunakan fungsi tersebut.

Jadi, dapat disimpulkan *Use case* adalah langkah – langkah atau urutan kegiatan yang dilakukan actor dan sistem informasi yang akan dibuat.Secara singkat, *Use case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah sistem

informasi dan siapa saja yang berhak menggunakan.

Tabel II.1. Simbol *Use Case*

Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar 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.
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apayang 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 : Sri Mulyani : 2016: 245)

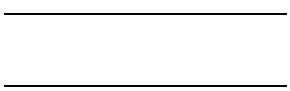
c. Diagram Aktivitas (*Activity Diagram*)

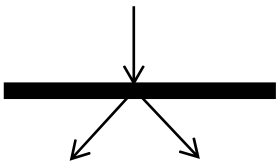
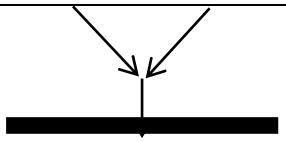
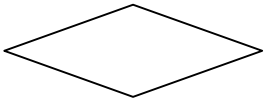
Activity diagram menggambarkan rangkaian aliran dari aktivitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti use case atau interaksi.

Menurut Sri Mulyani (2016 : 249) : "Activty diagram adalah diagram UML yang digunakan untuk menggambarkan alur aktivitas dari suatu proses" Menurut Muhammad Muslihudin dan Oktafianto (2016 :63) mengungkapkan: "Diagram aktivitas adalah tipe khusus dari diagram status yang memperlihatkan aliran dari suatu aktivitas ke aktivitas lainnya dalam suatu sistem".

Simbol-simbol yang digunakan dalam *activity diagram* dapat dilihat pada tabel II.2:

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.*
New Swimline	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

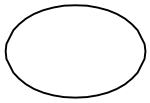
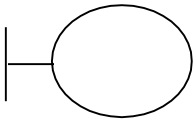
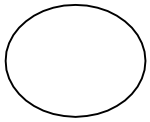
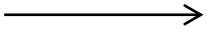
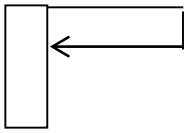
(Sumber : Sri Mulyani : 2016: 249)

d. Diagram Urutan (*Sequence Diagram*)

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek.

Menurut (Irmayani & Susyatih, 2017), *Sequence Diagram* menggambarkan bagaimana sistem merespon kegiatan user. *Sequence Diagram* yang dibuat yaitu berhubungan langsung dengna kegiatan utama dari sistem anggaran pendapatan dan belanja desa berbasis objek. Simbol-simbol yang digunakan dalam *sequence diagram* dapat dilihat pada tabel II.3:

Tabel II.3. Simbol *Sequence Diagram*

Gambar	Keterangan
	*EntityClass*, 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 sistem 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*.

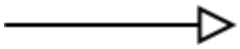
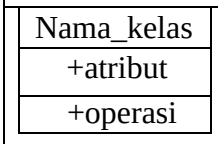
(Sumber : Irmayani & Susyati, 2017)

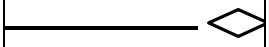
e. Diagram Kelas (*Class Diagram)*

Diagram kelas (*Class Diagram*) adalah diagram yang digunakan untuk menampilkan beberapa kelas serta paket-paket yang ada dalam sistem atau perangkat lunak yang sedang kita kembangkan. Dan berikut ini merupakan penjelasan mengenai class diagram:

Menurut Sri Mulyani (2016 : 247) menyatakan bahwa : "Class Diagram adalah diagram yang digunakan untuk mempresentasikan kelas, komponenkomponen kelas dan hubungan antara masing-masing kelas" Simbol *class diagram* dan *multiplicity class diagram* dapat dilihat pada Tabel II.4 dan TabelII.5.:

Tabel II.4. Simbol *Class Diagram*

Gambar	Keterangan
	*Generalization*, untuk menghubungkan antar kelas dengan arti umum-khusus. Jadi jika ada kelas dengan arti umum-khusus. Jadi jika ada kelas bermakna umum dan kelas bermakna khusus dapat menggunakan simbol ini.
	*Class*, untuk sebuah kelas pada struktur sistem. Penulisan tidak boleh menggunakan spasi. Simbol ini memiliki 3 susunan, yaitu kotak pertama adalah nama kelas, kedua atribut dan ketiga operasi.
	*Interface*, untuk simbol *interface* atau dalam bahasa indonesianya antar muka. Konsep yang digunakan pun sama dengan pemrograman berorientasi object (OOP).
	Association,digunakan untuk menghubungkan atau merelasikan kelas satu dengan kelas yang lainnya dengan makna umum.

	*Directed Association,* adalah relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain.
	*Aggregation,* adalah relasi antar kelas dengan makna semua bagian.
	Dependency, adalah relasi antar kelas dengan makna kebergantungan antar kelas.

(Sumber : Sri Mulyani : 2016 : 247)

Tabel II.5. ***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 : Sri Mulyani : 2016 : 247)