BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terdahulu

Adapun penelitian terkait yang akan digunakan sebagai sumber acuan yang relevan dan terkini yaitu:

Berdasarkan penelitian yang dilakukan oleh Dini et el (2017) mengenai "Perancangan Sistem Pakar Psikologi Untuk Diagnosis Gangguan Fobia", dari hasil analisis, perancangan dan implementasi, dapat disimpulkan bahwa telah berhasil dibangun sebuah sistem pakar yang dapat digunakan untuk mendiagnosa gangguan fobia dengan metode *forward chaining* yang mana metode tersebut mempunyai tingkatan prioritas pada aturan-aturan yang telah dimasukkan pada basis pengetahuan. Penelitian ini menghasilkan sistem pakar psikologi untuk diagnosis gangguan fobia yang bertujuan sebagai referensi dalam mengatasi gangguan fobia yang apabila diabaikan akan mengganggu kegiatan aktivitas sehari-hari.

Berdasarkan penelitian yang dilakukan oleh Hendi dan Andi (2020) mengenai "Sistem Pakar Diagnosa Gangguan Kecemasan Menggunakan Metode *Certainty Factor* Berbasis *Website*", diketahui bahwa gangguan kecemasan adalah suatu gangguan psikologis yang menyebabkan penderitanya mengalami rasa cemas yang besar dan berlebihan. Walaupun gangguan kecemasan ini sering dibicarakan dalam masyarakat, namun masih banyak yang tidak menyadari bahwa

mereka mengalami gangguan kecemasan. Salah satu cara untuk mengurangi dan membantu penderita gangguan kecemasan dalam mengetahui gejala, penyebab, atau penanggulangannya di bidang teknologi yaitu dengan menggunakan sistem pakar. Metode yang digunakan juga menggunakan metode *Certainty Factor*.

Berdasarkan penelitian yang dilakukan oleh Farajullah dan Murinto (2019) mengenai "Sistem Pakar Deteksi Dini Gangguan Kemasan Menggunakan Metode *Forward Chaining* Berbasis *Web*", telah berhasil dibuat suatu sistem yang berdasarkan pengetahuan seorang pakar yang ahli dibidangnya dengan memanfaatkan salah satu metode dalam bidang sistem pakar yakni *Forward Chaining*. Dari penelitian yang dilakukan menghasilkan sebuah perangkat lunak berbasis *web* untuk diagnosa penyakit gangguan kecemasan menggunakan metode *forward chaining* dengan kemampuan dapat memberikan informasi dalam mendiagnosa 15 data penyakit dan 16 data gejala gangguan kecemasan.

Berdasarkan penelitian yang dilakukan oleh Daniel et el (2021) mengenai "Sistem Pakar Untuk Mendiagnosa Gangguan Kesehatan Mental Menggunakan Algoritma Genetika", dapat disimpulkan bahwa dari pembuatan aplikasi sistem pakar ini dengan menggunakan metode faktor deterministik untuk mendiagnosa penyakit mental dapat membantu seseorang mendiagnosa jenis penyakit mental dengan gejala yang jelas tanpa pemeriksaan medis, dan memberikan solusi untuk diagnosisnya.

Berdasarkan penelitian yang dilakukan oleh Susanto, Cucut (2015) mengenai "Aplikasi Sistem Pakar Untuk Gangguan Mental Pada Anak Dengan

Metode *Certainty Factor*", dapat ditarik kesimpulan bahwa aplikasi sistem pakar untuk menentukan jenis gangguan mental pada anak dengan metode *Certainty Factor* yang dibangun dan dirancang dengan menggunakan bahasa pemrograman *Borland Delphi* 7.0 telah berhasil mendeteksi gangguan mental pada anak dengan sangat cepat.

II.2. Landasan Teori

Untuk mendukung keberhasilan penelitian ini, penyusun melakukan pendekatan teoritis melalui beberapa literatur yang berhubungan dengan penelitian yang dilakukan. Beberapa tinjauan pustaka pada penelitian ini yaitu:

II.2.1. Fobia

Menurut Gita dan Agus (2019), fobia adalah rasa takut yang tidak normal dan irasional (sulit dijelaskan alasannya) terhadap sesuatu (baik benda maupun situasi) secara berlebihan. Kata fobia sendiri berasal dari bahasa Yunani, yaitu "*phobos*" yang berarti "*fear*" (ketakutan). Fobia pertama kali diperkenalkan sebagai istilah kedokteran oleh Celsus, seorang romawi pencipta ensiklopedia. Meskipun beliau hidup dalam abad pertama S.M., namun kata fobia sendiri baru muncul dalam literatur psikiatri pada abad 19.

Menurut Yunita (2018), fobia termasuk ke dalam *psikoneurosis*, tetapi berbeda dengan gangguan kecemasan merata, gangguan fobia mengandung ketakutan yang cukup spesifik. Seseorang yang bereaksi dengan ketakutan yang amat sangat pada suatu stimulus atau situasi yang menurut kebanyakan orang lain tidaklah sangat berbahaya disebut orang yang mempunyai fobia. Fobia tidak

memiliki alasan yang irasional dan di luar kontrol si penderitanya. Banyak orang yang takut dengan ketinggian, tapi beberapa orang memiliki ketakutan yang berlebihan. Bahkan sebuah gambar atau video yang diambil dari suatu ketinggian seperti gedung pencakar langit dapat membuat penderita fobia ini mengalami peningkatan tekanan darah, jantung berdebar, dan peningkatan hormon kortisol.

Fobia bisa diderita siapa saja tanpa batasan usia dan jenis kelamin. Penderita fobia menyadari bahwa ketakutannya tidak beralasan dan berlebihan, namun ia sendiri tidak berdaya untuk mengatasinya. Pada tingkat yang ekstrim, penderita fobia akan merasa ia akan menjadi gila karena ketakutannya.

Berdasarkan buku panduan DSM-5, jenis-jenis fobia dapat digolongkan ke dalam 5 kategori, diantaranya:

1. Fobia situasional, yaitu tipe fobia terhadap suatu kondisi atau situasi seperti takut berada dalam pesawat terbang, lift, atau tempat tertutup.
2. Fobia hewan, yaitu tipe fobia terhadap binatang seperti tikus, anjing, atau binatang melata.
3. Fobia alam, yaitu tipe fobia terhadap lingkungan alam seperti ketinggian, kilat, atau air.
4. Fobia medis, yaitu tipe fobia terhadap darah, suntikan atau luka.
5. Fobia abstrak, yaitu tipe fobia terhadap hal-hal abstrak seperti kegagalan, cinta, kesendirian dan sebagainya.

II.2.2. Kecerdasan Buatan

Menurut Minda dan Sandra (2018), kecerdasan buatan atau *artificial intelligence* merupakan salah satu bagian dari ilmu komputer yang membuat agar mesin atau komputer dapat melakukan pekerjaan seperti dan sebaik manusia, dengan diberi bekal pengetahuan dan kemampuan untuk menalar.

Tujuan dari kecerdasan buatan ini yaitu membuat komputer lebih cerdas, mengerti tentang kecerdasan dan membuat mesin lebih berguna. Kecerdasan disini dimaksudkan dengan kemampuan untuk belajar atau mengerti dari pengalaman, mampu memahami pesan yang kontradiktif ataupun ambigu, mampu menanggapi dengan cepat dan baik atas situasi baru, dan menggunakan penalaran yang baik dalam memecahkan masalah serta menyelesaikannya dengan efektif.

II.2.3. Sistem Pakar

Salah satu cabang dari kecerdasan buatan adalah sistem pakar. Menurut Khairina et el (2018), sistem pakar (*expert system*) merupakan sistem yang berusaha untuk mengadopsi kemampuan atau pengetahuan manusia ke dalam komputer, agar komputer dapat bekerja dalam menyelesaikan suatu masalah seperti layaknya seorang pakar atau seseorang yang mempunyai keahlian dalam bidang tertentu.

Sebagai contohnya dokter adalah seorang pakar yang mampu mendiagnosis penyakit seorang pasien dan kemudian memberikan penjelasan tentang penyakit tersebut. Sistem pakar biasanya dianggap berhasil ketika sistem

pakar tersebut mampu mengambil keputusan seperti yang dilakukan oleh pakar aslinya baik dari sisi proses pengambilan keputusan maupun hasilnya.

Menurut Minda dan Sandra (2018), ada tiga orang yang terlibat dalam sistem pakar, yaitu:

1. Pakar, merupakan orang yang memiliki pengetahuan khusus, pendapat pengalaman dengan metode, serta kemampuan untuk mengaplikasi keahliannya tersebut guna menyelesaikan masalah.
2. *Knowledge engineer* (perekayasa sistem), adalah orang yang membantu pakar dalam menyusun area permasalahan dengan mengintegrasikan jawaban-jawaban pakar atas pertanyaan yang diajukan, menggambarkan analogi, mengajukan *counter example* dan menerangkan kesulitan-kesulitan konseptual.
3. Pemakai, sistem pakar memiliki beberapa pemakai, yaitu: pemakai bukan pakar, pelajar, pembangun sistem pakar yang ingin meningkatkan dan menambahkan basis pengetahuan, dan pakar.

II.2.4. *Certainty Factor*

Menurut Ahmad dan Hindayanti (2017), *Certainty Factor* (CF) diusulkan oleh Shortlife dan Buchanan pada tahun 1975 untuk mengakomodasi ketidakpastian pemikiran (*inexact reasoning*) seorang pakar. Seorang pakar, (misalnya dokter) sering menganalisis informasi yang ada dengan ungkapan seperti “mungkin”, “kemungkinan besar”, “hampir pasti”. Untuk mengakomodasi

hal ini kita menggunakan *Certainty Factor* (CF) guna menggambarkan tingkat keyakinan pakar terhadap masalah yang sedang dihadapi.

Untuk mendapatkan tingkat keyakinan (CF) dari sebuah *rule*, dapat dilakukan dengan mewawancarai seorang pakar. Nilai CF (*rule*) didapat dari interpretasi "*term*" dari pakar yang diubah menjadi CF tertentu.

Untuk menghitung nilai *Certainty Factor* dapat dilihat pada persamaan berikut:

CF[H,E] = MB[H,E] – MD [H,E]

Keterangan:

CF[H,E] : *Certainty Factor* hipotesa yang dipengaruhi oleh *evidence* E yang diketahui dengan pasti

MB[H,E] : *Measure of Belief* terhadap hipotesa H, jika diberikan *evidence* E (antara 0 dan 1)

MD : *Measure of Disbelief* (Nilai Ketidakpercayaan)

E : *Evidence* (Peristiwa/Fakta)

Untuk menghitung nilai CF_{gejala} dapat dilihat pada persamaan berikut:

$CF_{gejala} = CF_{pengguna} \times CF_{pakar}$

Keterangan:

CF_{gejala} : Nilai yang didapatkan dari hasil perkalian $CF_{pengguna}$ dan CF_{pakar}

$CF_{pengguna}$: Nilai jawaban dari pasien yang telah ditentukan

CF_{pakar} : Nilai standar pakar

Untuk menghitung nilai $CF_{gabungan}$ dapat dilihat pada persamaan berikut:

$CF_{gabungan} = CF_{lama} + CF_{gejala} \text{ x } (1 - CF_{lama})$

Keterangan:

$CF_{gabungan}$: $CF_{gabungan}$ digunakan apabila gejala lebih dari 1

CF_{lama} : nilai CF_{lama}

CF_{gejala} : Nilai CF berdasarkan gejala

Untuk menghitung $CF_{persentase}$ dapat dilihat pada persamaan berikut:

$CF_{persentase} = CF_{gabungan} \text{ x } 100\%$

Keterangan:

$CF_{gabungan}$: Besaran nilai $CF_{gabungan}$

$CF_{persentase}$: Nilai CF dalam persen (%)

II.2.5. XAMPP

XAMPP merupakan paket program *web* lengkap yang dapat dipakai untuk belajar pemograman *web*, khususnya PHP dan MySQL. XAMPP adalah perangkat lunak bebas, yang mendukung banyak sistem operasi dan merupakan kompilasi dari beberapa program. Fungsinya adalah sebagai server yang berdiri

sendiri (*localhost*), yang terdiri atas program *Apache HTTP Server*, database MySQL, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl (Chandra et ell, 2020).

II.2.6. UML

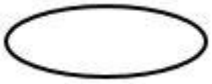
Menurut Suendri (2018), *Unified Modeling Language* (UML) adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak. UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem.

UML adalah sebuah bahasa yang berdasarkan grafik atau gambar untuk memvisualisasi, menspesifikasikan, membangun dan pendokumentasian dari sebuah sistem pengembangan *software*. Diagram pada *Unified Modeling Language* (UML) antara lain sebagai berikut:

1. Diagram *Use Case*

Diagram *use case* merupakan permodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih *actor* dengan sistem informasi yang akan dibuat. Dapat dikatakan *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 dapat dilihat pada tabel II.1. di bawah ini:

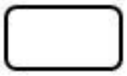
Tabel II.1. Simbol *Use Case*

Simbol	Nama	Keterangan
	*Actor*	Merupakan peran orang, sistem yang lain atau alat ketika berhubungan dengan *use case*.
	*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
	Association	Abstraksi dari penghubung antara aktor dengan *use case*.
	*Generalization*	Menunjukkan spesialisasi aktor untuk dapat berpartisipasi dengan *use case*.
<<include>> 	*Include*	Menunjukkan bahwa suatu *use case* seluruhnya merupakan fungsionalitas dari *use case* lainnya.
<<extend>> 	*Extend*	Menunjukkan bahwa suatu *use case* merupakan tambahan fungsional dari *use case* lainnya jika suatu kondisi terpenuhi.

2. Diagram Aktivitas (*Activity Diagram*)

Diagram aktivitas menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram* dapat dilihat pada tabel II.2. berikut:

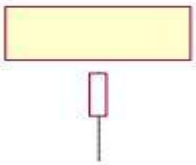
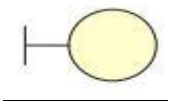
Tabel II.2. Simbol *Activity Diagram*

Simbol	Nama	Keterangan
	Status awal	Sebuah diagram aktivitas memiliki sebuah status awal.
	Aktivitas	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
	*Decision*/percabangan	Percabangan di mana ada pilihan aktivitas yang lebih dari satu.
	Join/penggabungan	Penggabungan yang mana lebih dari satu aktivitas lalu digabungkan jadi satu.
	Status akhir	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
	*Swimlane*	*Swimlane* memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

3. 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. Simbol-simbol yang digunakan dalam *sequence diagram* dapat dilihat pada tabel II.3. berikut:

Tabel II.3. Simbol *Sequence Diagram*

Simbol	Nama	Keterangan
	*Actor*	Mempresentasikan entitas yang berada di luar sistem dan berinteraksi dengan sistem.
	Lifeline	Menghubungkan objek selama *sequence* (*message* dikirim atau diterima dan aktivasinya).
	*General*	Mempresentasikan entitas tunggal dalam *sequence diagram*.
	*Boundary*	Berupa tepi dari sistem, seperti *user interface* atau suatu alat yang berinteraksi dengan sistem yang lain.
	*Control*	Elemen yang mengatur aliran dari informasi untuk sebuah skenario. Objek ini umumnya mengatur

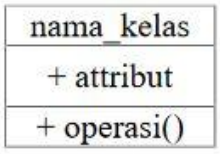
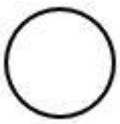
		perilaku bisnis.
	Entitas	Elemen yang bertanggung jawab menyimpan data atau informasi. Ini dapat berupa model objek.
	*Activation*	Suatu titik di mana objek mulai berpartisipasi di dalam sebuah *sequence* yang menunjukkan kapan sebuah objek mengirim atau menerima objek.
	Message	Berfungsi sebagai komunikasi antar objek yang mengambarkan aksi yang akan dilakukan.
	Message Entry	Berfungsi untuk menggambarkan pesan/hubungan antar objek yang menunjukkan urutan kejadian yang terjadi.
	*Message to Self*	Simbol ini menggambarkan pesan/hubungan objek itu sendiri, yang menunjukkan urutan kejadian yang terjadi.
	*Message Return*	Menggambarkan hasil dari pengiriman *message* dan digambarkan dengan arah dari kanan

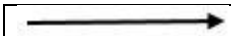
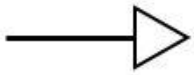
		dan ke kiri.

4. Diagram Kelas (*Class Diagram*)

Class diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations*/*Method*), *Visibility*, tingkat akses objek external pada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti. Simbol *class diagram* dan *multiplicity class diagram* dapat dilihat pada tabel II.4. dan tabel II.5. di bawah ini:

Tabel II.4. Simbol *Class Diagram*

Simbol	Nama	Keterangan
	*Class*	Kelas pada struktur sistem.
	*Interface*	Sama dengan konsep *interface* dalam pemrograman berorientasi objek.
	Association	Relasi antara *class* dengan arti umum, asosiasi biasanya juga disertai dengan *Multiplicity*.
	Directed Association	Relasi antarkelas dengan makna

		kelas yang digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*.
	Generalisasi	Relasi antar kelas dengan makna generalisasi spesialisasi.
	Dependency	Relasi antar kelas dengan makna ketergantungan antarkelas.
	*Aggregation*	Relasi antarkelas dengan makna semua-bagian (*whole-part*).

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