**SKRIPSI**

**DIAGNOSA PENYAKIT PADA AYAM PETELUR MENGGUNAKAN METODE *CERTAINTY FACTOR***

****

**Oleh :**

# HALAMAN JUDUL

**JURUSAN INFORMATIKA**

**FAKULTAS SAINS DAN TEKNOLOGI**

**UNIVERSITAS PERADABAN**

**2024**

# 

# PERSETUJUAN SKRIPSI

JUDUL : DIAGNOSA PENYAKIT PADA AYAM PETELUR   
 MENGGUNAKAN METODE *CERTAINTY FACTOR*

NAMA :

NIM :

**Skripsi ini telah disetujui untuk diujikan di hadapan**

**Dewan Penguji Skripsi**

PADA : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| Dosen Pembimbing 1,  **Khurotul Aeni, M. Kom**  NIDN. 0618098802 | Dosen Pembimbing 2,  **Nurul Mega Saraswati, M.Kom**  NIDN. 0606069102 |
| Ka. Jurusan Informatika  **Khurotul Aeni, M. Kom**  NIDN. 0618098802 | |

# PENGESAHAN SKRIPSI

JUDUL : DIAGNOSA PENYAKIT PADA AYAM PETELUR   
 MENGGUNAKAN METODE *CERTAINTY FACTOR*

NAMA :

NIM :

**Skripsi ini telah diseminarkan dihadapan Dewan Penguji pada Sidang Skripsi tanggal \_\_\_\_\_\_\_\_\_\_**

Bumiayu, \_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| Menyetujui, | |
| Dosen Penguji 1, | Dosen Penguji 2, |
| Dosen Pembimbing 1,  **Khurotul Aeni, M. Kom**  NIDN. 0618098802 | Dosen Pembimbing 2,  **Nurul Mega Saraswati, M.Kom**  NIDN. 0606069102 |
| Mengetahui, | |
| Ka. Jurusan Informatika  **Khurotul Aeni, M. Kom**  NIDN. 0618098802 | |

# DAFTAR ISI

[HALAMAN JUDUL 1](#_Toc173548553)

[PERSETUJUAN SKRIPSI ii](#_Toc173548554)

[PENGESAHAN SKRIPSI iii](#_Toc173548555)

[DAFTAR ISI iv](#_Toc173548556)

[DAFTAR TABEL vi](#_Toc173548557)

[DAFTAR GAMBAR vii](#_Toc173548558)

[BAB I PENDAHULUAN 1](#_Toc173548559)

[1.1 Latar Belakang 1](#_Toc173548560)

[1.2 Rumusan Masalah 2](#_Toc173548561)

[1.3 Batasan Masalah 2](#_Toc173548562)

[1.4 Tujuan Penelitian 3](#_Toc173548563)

[1.5 Manfaat Penelitian 3](#_Toc173548564)

[1.6 Sistematika Penulisan 4](#_Toc173548565)

[BAB II TINJAUAN PUSTAKA 5](#_Toc173548566)

[2.1 Penelitian Terkait 5](#_Toc173548567)

[2.2 Landasan Teori 6](#_Toc173548568)

[2.2.1 Ayam Petelur 6](#_Toc173548569)

[2.2.2 Penyakit Pada Ayam 7](#_Toc173548570)

[2.2.3 Sistem Pakar 9](#_Toc173548571)

[2.2.4 *Certainly Factor* 12](#_Toc173548572)

[2.2.5 *Unified Modeling Language (UML*) 14](#_Toc173548573)

[2.2.7 *Blackbox testing* 22](#_Toc173548574)

[BAB III METODOLOGI PENELITIAN 23](#_Toc173548575)

[3.1 Tahap Penelitian 23](#_Toc173548576)

[3.2 Metode Pengumpulan data 23](#_Toc173548577)

[3.3 Metode Pengembangan Sistem 23](#_Toc173548578)

[3.4 Metode yang diusulkan 24](#_Toc173548579)

[3.4. 1 Analisis 24](#_Toc173548580)

[3.4. 2 Desain 31](#_Toc173548581)

[3.4. 3 Pengkodean 31](#_Toc173548582)

[3.4. 4 Testing 31](#_Toc173548583)

[BAB IV HASIL DAN PEMBAHASAN 32](#_Toc173548584)

[4.1 Construction (Code & Test) 32](#_Toc173548585)

[4.2 Implementasi Perangkat *Hardware* dan *Software* 32](#_Toc173548586)

[4.3 Struktur Table 32](#_Toc173548587)

[4.4 Implementasi Antarmuka 33](#_Toc173548588)

[4.5 Pengujian 37](#_Toc173548589)

[BAB V KESIMPULAN 40](#_Toc173548590)

[5. 1 Kesimpulan 40](#_Toc173548591)

[5. 2 Saran 40](#_Toc173548592)

[DAFTAR PUSTAKA 32](#_Toc173548593)

# DAFTAR TABEL

[Tabel 3. 1 Data Penyakit Ayam 25](#_Toc173548506)

[Tabel 3. 2 Gejala Penyakit pada ayam 26](#_Toc173548507)

[Tabel 3. 3 Rules 28](#_Toc173548508)

[Tabel 4. 1 Pengujian Kelola Data Penyakit 37](#_Toc173548528)

[Tabel 4. 2 Pengujian Kelola Data Gejala 38](#_Toc173548529)

[Tabel 4. 3 Pengujian Kelola Data Rule 38](#_Toc173548530)

[Tabel 4. 4 Pengujian Perhitungan 39](#_Toc173548531)

[Tabel 4. 5 Pengujian Diagnosa 39](#_Toc173548532)

# DAFTAR GAMBAR

[Gambar 2. 1 Komponen Sistem Pakar 12](#_Toc173548393)

[Gambar 3. 1 Tahap Penelitian 23](#_Toc173548399)

[Gambar 3. 2 Linear Squence Model 24](#_Toc173548400)

[Gambar 4. 1 Struktur Tabel penyakit 32](#_Toc173548470)

[Gambar 4. 2 Struktur tabel Gejala 33](#_Toc173548471)

[Gambar 4. 3 Struktur tabel rule 33](#_Toc173548472)

[Gambar 4. 4 Halaman Dashboard 33](#_Toc173548473)

[Gambar 4. 5 Halaman Data Penyakit 34](#_Toc173548474)

[Gambar 4. 6 Halaman Data Gejala 34](#_Toc173548475)

[Gambar 4. 7 Halaman Data Rule 35](#_Toc173548476)

[Gambar 4. 8 Halaman Implementasi 35](#_Toc173548477)

[Gambar 4. 9 Halaman Perhitungan 36](#_Toc173548478)

[Gambar 4. 10 Halaman Diagnosa 36](#_Toc173548479)

[Gambar 4. 11 Halaman Proses Diagnosa 37](#_Toc173548480)

# BAB I PENDAHULUAN

1. **Latar Belakang**

Ayam petelur merupakan salah satu komoditas penting dalam industri peternakan di Indonesia. Telur ayam menjadi sumber protein hewani yang mudah dijangkau dan diolah oleh masyarakat. Menurut data Direktorat Jenderal Peternakan dan Kesehatan Hewan (Ditjen PKH) Kementerian Pertanian, produksi telur ayam nasional pada tahun 2023 mencapai 22,1 juta ton [1]. Kesehatan ayam petelur menjadi faktor penting dalam menjaga produktivitas telur. Penyakit pada ayam petelur dapat menyebabkan penurunan produksi telur, bahkan kematian ayam. Hal ini dapat berakibat pada kerugian ekonomi bagi peternak dan berkurangnya ketersediaan telur di pasaran[2].

Diagnosis penyakit pada ayam petelur seringkali menjadi tantangan bagi peternak. Gejala penyakit pada ayam petelur dapat bervariasi dan sulit dibedakan. Selain itu, ketersediaan dokter hewan yang kompeten di daerah pedesaan masih terbatas. Hal ini menyebabkan keterlambatan dalam diagnosis dan pengobatan penyakit, yang dapat memperparah kondisi ayam dan meningkatkan kerugian ekonomi bagi peternak. Metode Certainty Factor (CF) merupakan salah satu metode kecerdasan buatan yang dapat digunakan untuk mendiagnosis penyakit. CF menggunakan nilai numerik untuk mewakili tingkat kepastian diagnosis. Nilai CF yang lebih tinggi menunjukkan tingkat kepastian yang lebih tinggi .

Penerapan metode CF dalam diagnosis penyakit ayam petelur menunjukkan hasil yang menjanjikan. Penelitian mengenai Sistem Pakar Diagnosa Penyakit pada Ayam Dengan Metode *Certainty Factor* pernah dilakukan, sistem yang dibangun berbasis android dan berhasil mendiagnosa penyakit pada ayam berdasarkan gejala-gejala yang timbul pada ayam. Berdasarkan hasil uji *blackbox* sistem pakar yang dibuat dapat berjalan dengan baik. Sedangkan berdasarkan pengujian UAT dengan nilai 96,2% dapat diambil kesimpulan bahwa sistem pakar bisa diterima dan digunakan oleh masyarakat [3]. Penelitian lain mengenai Sistem Pakar Diagnosis Penyakit Pada Ayam Menggunakan Metode *Certainty Factor* Secara fungsional sistem pakar ini dapat berjalan dengan baik setelah dilakukan pengujian menggunakan metode *Black-Box testing*. Dan pengujian yang ditujukan terhadap pakar didapatkan hasil presentase dari 2 penyakit dengan 10 gejala adalah 100% sedangnkan di sitem pakar diagnosa penyakit ayam menunjukan presentase 67,744% pada penyakit gumbro dengan jumlah 5 gejala dan presentase 69,706% pada penyakit mareks dengan jumlah 5 gejala [4]. Beberapa penelitian telah dilakukan untuk mengembangkan sistem pakar diagnosis penyakit ayam petelur menggunakan metode CF. Penelitian-penelitian tersebut menunjukkan bahwa metode CF dapat digunakan secara efektif untuk mendiagnosis penyakit ayam petelur dengan tingkat akurasi yang tinggi.

Penelitian ini bertujuan untuk mengembangkan sistem pakar diagnosis penyakit ayam petelur menggunakan metode CF yang lebih akurat dan mudah digunakan oleh peternak. Sistem pakar ini akan dilengkapi dengan fitur-fitur yang lebih lengkap, seperti database penyakit yang lebih luas, antarmuka yang lebih *user-friendly*, dan kemampuan untuk memberikan saran pengobatan yang tepat. Penelitian ini diharapkan dapat membantu peternak dalam mendiagnosis penyakit pada ayam petelur secara lebih akurat dan tepat waktu. Hal ini dapat membantu meningkatkan produktivitas telur ayam dan mengurangi kerugian ekonomi bagi peternak. Selain itu, penelitian ini juga diharapkan dapat membantu meningkatkan ketersediaan telur di pasaran dan memenuhi kebutuhan protein hewani bagi masyarakat, Sehingga penulis memilih judul “**Diagnosa Penyakit Pada Ayam Petelur Menggunakan Metode *Certainty Factor”.***

1. **Rumusan Masalah**

Berdasarkan latar belakang di atas maka rumusan masalah pada penelitian ini adalah berapa tingkat akurasi Metode *Certainly Factor* untuk melakukan diagnosis penyakit ayam petelur? “

1. **Batasan Masalah**

Penelitian ini dibuat dengan batasan masalah agar lebih terfokus pada penyelesaian masalah yang telah ditentukan sebelumnya, dengan demikian, hasil yang diperoleh dari penelitian ini diharapkan dapat memberikan solusi yang lebih spesifik dan bermanfaat bagi masalah yang sedang diteliti. Batasan masalah sebagai berikut:

1. Data Penelitian

Data penyakit, gejala dan pengetahuan pakar yang digunakan dalam penelitian ini bersumber dari jurnal rujukan lain.

1. Validasi Sistem

Sistem pakar yang dikembangkan dalam penelitian ini divalidasi dengan menggunakan data yang berasal dari jurnal rujukan dan menggunakan metode validasi *self-validation.*

1. **Tujuan Penelitian**

Tujuan dari penelitian ini adalah untuk mengetahui tingkat akurasi metode *Certainly Factor* untuk diagnosis penyakit ayam petelur.

1. **Manfaat Penelitian**

Manfaat Penelitian Penggunaan *Metode Certainty Factor* untuk Diagnosis Penyakit Ayam Petelur. Penelitian ini diharapkan dapat memberikan beberapa manfaat, di antaranya**:**

**Untuk Universitas:**

1. **Meningkatkan Reputasi Akademik:** Universitas yang berpartisipasi dalam penelitian ini akan dikenal sebagai institusi yang berkontribusi pada inovasi teknologi dan solusi praktis dalam bidang peternakan.
2. **Menambah Literatur Ilmiah:** Penelitian ini akan menambah jumlah publikasi ilmiah yang dihasilkan oleh universitas, yang dapat digunakan sebagai referensi bagi peneliti lain di masa depan.
3. **Memperluas Jaringan Kolaborasi:** Universitas dapat memperluas kerjasama dengan industri peternakan dan lembaga penelitian lainnya, baik di dalam maupun luar negeri.

**Untuk Peneliti:**

1. **Mengembangkan Keahlian dan Pengetahuan:** Peneliti akan mendapatkan kesempatan untuk mengembangkan pengetahuan dan keterampilan mereka dalam penerapan metode Certainty Factor dan diagnosis penyakit pada ayam petelur.
2. **Peluang Publikasi dan Presentasi:** Hasil penelitian ini berpotensi untuk dipublikasikan di jurnal-jurnal ilmiah terkemuka dan dipresentasikan di konferensi-konferensi nasional maupun internasional.
3. **Meningkatkan Kapasitas Riset:** Penelitian ini akan meningkatkan kapasitas dan pengalaman peneliti dalam melakukan riset yang aplikatif dan berdampak nyata pada industri peternakan.

**Untuk Peternak:**

1. **Meningkatkan Akurasi Diagnosis:** Peternak dapat memanfaatkan metode Certainty Factor untuk meningkatkan akurasi diagnosis penyakit pada ayam petelur, sehingga tindakan pengobatan dapat dilakukan lebih tepat.
2. **Membantu Pengambilan Keputusan:** Dengan diagnosis yang lebih akurat, peternak dapat membuat keputusan pengobatan yang lebih baik dan tepat waktu, yang akan meningkatkan kesehatan dan produktivitas ayam petelur.
3. **Efisiensi Waktu dan Biaya:** Metode ini diharapkan dapat menghemat waktu dan biaya dalam proses diagnosis, yang pada akhirnya dapat meningkatkan efisiensi operasional peternakan.
4. **Meningkatkan Keuntungan:** Dengan mengurangi kerugian akibat penyakit dan meningkatkan produktivitas ayam, peternak dapat meningkatkan keuntungan dari usaha peternakan mereka.
5. **Sistematika Penulisan**

Penyusunan proposal skripsi ini diuraikan dalam beberapa sub bab yang tersusun sebagai berikut:

**BAB I PENDAHULUAN**

Bab ini membahas mengenai latar belakang, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

**BAB II TINJAUAN PUSTAKA**

Pada bab ini menjelaskan tentang teori-teori dasar yang berhubungan dengan penelitian.

**BAB III METODOLOGI PENELITIAN**

Pada bab ini membahas tentang teori-teori dan metodologi penelitian yang digunakan.

**BAB IV HASIL DAN PEMBAHASAN**

Pada bab ini akan membahan hasil dan pembahasan pada penelitian.

**BAB V KESIMPULAN**

Pada bab ini akan memaparkan kesimpulan dari penelitian.

# 

# BAB II TINJAUAN PUSTAKA

1. **Penelitian Terkait**

Beberapa penelitian terdahulu telah dilakukan oleh peneliti sebelumnya. Berikut beberapa penelitian terkait diagnosis penyakit pada ayam petelur menggunakan metode *certainty factor*. Penelitian mengenai Sistem Pakar Diagnosis Penyakit pada Ayam Petelur Menggunakan Metode Certainty Factor Berbasis Android Penelitian ini mengembangkan sistem pakar berbasis web untuk mendiagnosis penyakit pada ayam petelur menggunakan metode *certainty factor*. Sistem ini menunjukkan akurasi diagnosis mencapai 96,2% [3].

Penelitian selanjutnya mengenai Implementasi *Certainty Factor* untuk Diagnosis Penyakit pada Ayam Petelur. Penelitian ini mengimplementasikan metode *certainty factor* pada sistem pakar untuk mendiagnosis penyakit pada ayam petelur. Hasil pengujian menunjukkan bahwa sistem pakar ini memiliki tingkat validitas sebesar 88,67% [4].

Penelitian selanjutnya mengenai Sistem Pakar untuk Diagnosis Penyakit Ayam menggunakan Metode *Certainty Factor* (Studi Kasus: Balai Besar Pelatihan Peternakan Batu) hasil dari penelitian ini memiliki tingkat akurasi paling rendah yang didapatkan dengan menggunakan sistem pakar metode *Certainty Factor* adalah 88% dalam 10 pengujian tingkat akurasi [5].

Selain penelitian-penelitian di atas, masih banyak penelitian lain yang terkait dengan diagnosis penyakit pada ayam petelur menggunakan metode *certainty factor*. Metode ini terbukti efektif dalam membantu peternak ayam untuk mendiagnosis penyakit pada ayam mereka dengan cepat dan akurat.

**Table 2. 1 Daftar Penelitian Terdahulu**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Penulis** | **Tahun** | **Judul Penelitian** | **Metode** | **Hasil** |
| 1 | Tengku Khairil Ahsyar, Tio Doli Raharjo, Syaifullah | 2021 | Sistem Pakar Diagnosa Penyakit Pada Ayam Dengan Metode *Certainty Factor* Berbasis Android | *Certainty Factor* | Hasil dari penlitian ini adalah membuat aplikasi berbasis android dengan akurasi uji sebesar 96,2% |
| 2 | Ishak Panangian Sinaga, Arief Andy Soebroto, Imam Cholissodin | 2022 | Sistem Pakar untuk Diagnosis Penyakit Ayam menggunakan *Metode Certainty Factor* (Studi Kasus: Balai Besar Pelatihan Peternakan Batu) | *Certainty Factor* | Hasil dari penlitian ini adalah membuat aplikasi berbasis android dengan akurasi uji sebesar 92,8% |
| 3 | Jeremias Febronius Bere, Joseph Dedy Irawan, F.X. Ariwibisono | 2021 | Sistem Pakar Diagnosis Penyakit Pada Ayam Menggunakan Metode *Certainty Factor* | *Certainty Factor* | Diagnosa penyakit ayam menunjukan presentase 67,744% pada penyakit gumbro dengan jumlah 5 gejala dan presentase 69,706% pada penyakit mareks dengan jumlah 5 gejala. |
| 4 | Salsabila Putri Rajani Said, I Gede Pasek Suta Wijaya, Fitri Bimantoro | 2020 | Implementasi Metode *Dempster Shafer* pada Sistem Pakar Diagnosis Penyakit Ayam (*Implementation of Dempster Shafer Method on Expert System of Diagnosing Chicken Disease*) | *Dempster Shafer* | Hasil pengujian akurasi pada 30 contoh kasus yang telah dilakukan menghasilkan kesesuaian tingkat akurasi sebesar 92,22% |
| 5 | Faustino L.M Lalo Nusa, Suryo Adi Wibowo, Deddy Rudhistiar | 2022 | Sistem Pakar Diagnosis Penyakit Pada Anjing Menggunakan Metode *Certainty Factor* |  | Hasil akurasi menunjukkan tingkat 82,21888% pada rabies dengan 7 efek samping dan 69,706% pada skabies dengan 5 indikasi, Secara fungsional sistem pakar ini dapat berjalan dengan baik setelah dilakukan pengujian menggunakan metode Black- Box testing untuk melakukan pengujian dimana sistem berjalan sesuai dengan harapan/sesuai dengan tujuan ketika sistem pakar diagnosis penyakit pada ayam dijalankan. |

Berdasarkan Tabel 2.1, terlihat bahwa penelitian ini tidak memiliki perbedaan yang signifikan dengan penelitian-penelitian sebelumnya. Hal ini mengindikasikan bahwa topik yang diteliti telah banyak dibahas dan dianalisis oleh peneliti-peneliti terdahulu. Namun demikian, penelitian ini tetap memiliki kontribusi dan keunikan tersendiri diantaranya proses *building* aplikasi menggunakan teknologi versi terbaru yang dapat mengindentifikasikan aplikasi menyesuaikan dengan teknologi versi terbaru, Studi kasus yang digunakan pada penelitian ini juga berbeda dengan penelitian lainnya.

1. **Landasan Teori**
2. **Ayam Petelur**

Ayam petelur didefinisikan sebagai ayam betina dewasa yang dipelihara secara khusus untuk diambil telurnya. Ayam petelur modern memiliki potensi produksi telur lebih dari 500 butir dalam 700 hari (100 minggu). Produksi telur mencapai puncaknya pada usia 140 hari dengan tingkat produksi mencapai 98% . Ayam petelur diklasifikasikan menjadi dua, yaitu ayam petelur ringan dan ayam petelur medium . Ciri-ciri Ayam Petelur yaitu:

1. Ukuran tubuh relatif kecil dan ramping
2. Cepat dewasa kelamin
3. Tingkah laku lincah
4. Mudah terkejut
5. Sensitif terhadap stress
6. Efisien dalam mengolah zat-zat makanan menjadi telur.

Ciri-ciri ayam petelur ringan:

1. Badan ramping, kurus, mungil, dan kecil
2. Mata bersinar
3. Bulu berwarna putih bersih
4. Jengger merah
5. Telur berwarna putih kecoklatan
6. Berat badan dewasa sekitar 1,7 kg
7. Contoh ras: Isa Brown, Lohmann Brown, Hy-Line Brown

Ciri-ciri ayam petelur medium:

1. Badan lebih besar dan kokoh dibandingkan ayam petelur ringan
2. Bulu berwarna cokelat muda atau kemerahan
3. Jengger merah
4. Telur berwarna cokelat
5. Berat badan dewasa sekitar 2,1 kg
6. Contoh ras: Rhode Island Red, Australorp, Plymouth Rock

Faktor-faktor yang memengaruhi produksi telur:

1. Usia ayam
2. Genetik ayam
3. Pakan
4. Kualitas air minum
5. Pengelolaan kandang
6. Kondisi kesehatan ayam
7. Stres
8. **Penyakit Pada Ayam**

Ayam merupakan salah satu jenis unggas yang umum dipelihara di Indonesia, baik untuk diambil dagingnya maupun telurnya. Namun, ayam pun tak luput dari berbagai penyakit yang dapat menyerang dan menyebabkan kerugian bagi peternak.

1. **Definisi Penyakit pada Ayam**

Penyakit pada ayam adalah kondisi abnormal yang disebabkan oleh agen infeksius, faktor lingkungan, atau kombinasi keduanya, yang mengganggu fungsi fisiologis normal ayam dan menyebabkan penurunan produktivitas, kesakitan, dan kematian.

Penyakit pada ayam dapat diklasifikasikan berdasarkan berbagai faktor, seperti:

* **Penyebab:** virus, bakteri, jamur, parasit, protozoa, dan lain-lain.
* **Lokasi:** sistem pernapasan, pencernaan, saraf, kulit, dan lain-lain.
* **Dampak:** akut, kronis, menular, dan tidak menular.

1. **Faktor Risiko Penyakit pada Ayam**

Beberapa faktor yang dapat meningkatkan risiko ayam terserang penyakit, antara lain:

* **Usia:** ayam muda lebih rentan terhadap penyakit dibandingkan ayam dewasa.
* **Stres:** stres dapat melemahkan sistem kekebalan tubuh ayam dan membuatnya lebih mudah terserang penyakit.
* **Lingkungan:** lingkungan yang kotor, lembab, dan padat dapat menjadi tempat berkembang biaknya agen penyakit.
* **Manajemen:** manajemen pemeliharaan yang buruk, seperti sanitasi yang tidak memadai dan pemberian pakan yang tidak seimbang, dapat meningkatkan risiko penyakit.

1. **Dampak Penyakit pada Ayam**

Penyakit pada ayam dapat menyebabkan berbagai dampak negatif, antara lain:

* **Penurunan produktivitas:** ayam yang sakit umumnya mengalami penurunan nafsu makan dan produksi telur atau daging.
* **Kematian:** beberapa penyakit pada ayam dapat berakibat fatal dan menyebabkan kematian.
* **Kerugian ekonomi:** kematian ayam, biaya pengobatan, dan penurunan produktivitas dapat menyebabkan kerugian ekonomi yang signifikan bagi peternak.

1. **Pencegahan dan Pengendalian Penyakit pada Ayam**

Pencegahan dan pengendalian penyakit pada ayam sangat penting untuk menjaga kesehatan ayam dan meminimalkan kerugian ekonomi bagi peternak. Beberapa langkah pencegahan dan pengendalian yang dapat dilakukan, antara lain:

* **Vaksinasi:** pemberian vaksin yang sesuai dapat melindungi ayam dari berbagai penyakit.
* **Biosecurity:** penerapan biosecurity yang ketat dapat mencegah masuknya agen penyakit ke dalam kandang ayam.
* **Sanitasi:** menjaga kebersihan kandang dan lingkungan sekitar kandang secara rutin.
* **Manajemen pemeliharaan yang baik:** memberikan pakan yang seimbang, menjaga kepadatan kandang yang ideal, dan memantau kesehatan ayam secara berkala.

**Pengobatan:** jika ayam terserang penyakit, segera lakukan pengobatan yang tepat sesuai dengan diagnosis dokter hewan.

1. **Sistem Pakar**
2. Pengertian Sistem Pakar

sistem pakar (*expert system*) merupakan sebuah program komputer yang dirancang untuk memodelkan kemampuan menyelesaikan masalah-masalah spesifik seperti layaknya seorang pakar (*humam expert*). Seorang pakar adalah seorang individu yang memiliki kemampuan pemahaman yang superior dari suatu masalah, contohnya seorang dokter, penasihat keuangan, pakar mesin mobil, teknisi komputer, dan sebagainya. Berbagai sistem pendukung keputusan klinis seperti halnya sistem pakar dapat digunakan untuk memfasilitasi diagnosis medis. Sistem pakar memiliki arsitektur, komponen, dan fungsi yang apat meningkatkan kemampuan membuat keputusan secara keseluruhan untuk mendiagnosis penyakit [6]. Sistem pakar dapat dibagi menjadi tiga komponen terpisah yaitu, *knowledge base*, *obcervation base*, dan *inference base*. Sistem pakar biasa disebut dengan knowledge-based system. Sistem in

1. Tujuan dan Manfaat Sistem Pakar

**Sistem pakar memiliki beberapa tujuan utama, yaitu:**

1. Membantu pakar dalam menyelesaikan masalah yang kompleks: SP dapat membantu pakar dalam menyelesaikan masalah yang kompleks dengan lebih cepat dan efisien
2. Menyediakan akses ke pengetahuan pakar: SP dapat menyediakan akses ke pengetahuan pakar bagi orang-orang yang tidak memiliki keahlian di bidang tersebut.
3. Melestarikan pengetahuan pakar: SP dapat membantu melestarikan pengetahuan pakar yang mungkin hilang seiring waktu.

**Manfaat penggunaan sistem pakar antara lain:**

1. Meningkatkan efisiensi dan efektivitas: SP dapat membantu menyelesaikan masalah dengan lebih cepat dan akurat, sehingga meningkatkan efisiensi dan efektivitas.
2. Meningkatkan kualitas keputusan: SP dapat membantu dalam membuat keputusan yang lebih baik dengan menyediakan informasi dan saran yang komprehensif.
3. Mengurangi biaya: SP dapat membantu mengurangi biaya dengan mengotomatisasi tugas-tugas yang kompleks dan memakan waktu.
4. Arsitektur Sistem Pakar

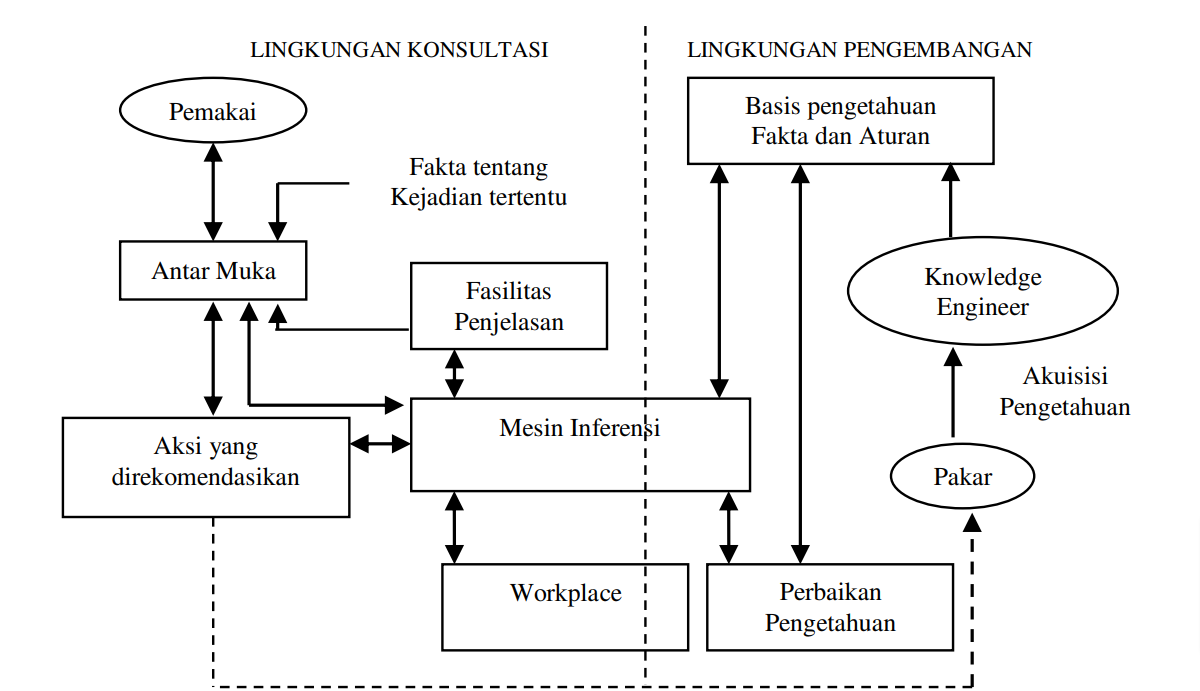
Arsitektur sistem pakar umumnya terdiri dari beberapa komponen utama, yaitu:

1. Basis pengetahuan: Menyimpan fakta, aturan, dan hubungan antar konsep dalam domain permasalahan.
2. Mesin inferensi: Melakukan penalaran berdasarkan fakta dan aturan yang terdapat dalam basis pengetahuan untuk menghasilkan kesimpulan atau saran.
3. Subsistem antarmuka pengguna: Memungkinkan pengguna untuk berinteraksi dengan sistem pakar dan mendapatkan informasi atau saran yang dibutuhkan.
4. Subsistem akuisisi pengetahuan: Membantu dalam membangun dan memelihara basis pengetahuan sistem pakar.
5. Subsistem penjelasan: Menjelaskan alasan di balik kesimpulan atau saran yang diberikan oleh sistem pakar.
6. Pengembangan Sistem Pakar

Pengembangan sistem pakar umumnya dilakukan melalui beberapa tahap, yaitu:

1. Analisis kebutuhan: Mengidentifikasi kebutuhan dan tujuan dari sistem pakar.
2. Akuisisi pengetahuan: Mengumpulkan dan merepresentasikan pengetahuan pakar dalam bentuk basis pengetahuan.
3. Pengembangan basis pengetahuan: Membangun dan memelihara basis pengetahuan sistem pakar.
4. Pengembangan mesin inferensi: Memilih dan mengembangkan mesin inferensi yang sesuai dengan kebutuhan sistem pakar.
5. Pengembangan antarmuka pengguna: Merancang dan membangun antarmuka pengguna yang mudah digunakan dan dipahami.
6. Pengujian dan evaluasi: Menguji dan mengevaluasi sistem pakar untuk memastikan bahwa sistem tersebut berfungsi dengan benar dan memenuhi kebutuhan pengguna.
7. Komponen system pakar

Sistem pakar disusun oleh dua bagian utama, yaitu bagian lingkungan pengembangan (development environment) dan lingkungan konsultasi (consultation environment). Lingkungan pengembang digunakan oleh pembuat sistem pakar untuk membangun komponen-komponennya dan memperkenalkan pengetahuan ke dalam knowledge base (basis pengetahuan). Sedangkan lingkungan konsultasi digunakan oleh pengguna untuk berkonsultasi sehingga pengguna mendapatkan pengetahuan dan nasihat dari sistem pakar layaknya berkonsultasi dengan seorang pakar. Komponen-komponen sistem pakar .



Gambar 2. 1 Komponen Sistem Pakar

1. *Knowledge Base* (Basis Pengetahuan)

Basis pengetahuan merupakan hasil akuisisi dan representasi pengetahuan dari seorang pakar. Basis pengetahuan berisi pengetahuan-pengetahuan dalam penyelesaian masalah. Basis pengetahuan terdiri dari dua elemen dasar yaitu fakta dan rule atau aturan.

1. *Inference Engine* (Mesin Inferensi)

Mesin inferensi adalah sebuah program yang berisi metodologi yang digunakan untuk melakukan penalaran terhadap informasi-informasi dalam basis pengetahuan untuk memformulasikan konklusi.

1. *User Interface* (Antar Muka Pengguna)

User interface adalah penghubung antar program sistem pakar dengan pengguna yang dapat dihubungkan melalui website, dekstop ataupun mobile. Antarmuka digunakan sebagai media komunikasi antara pengguna dan sistem pakar.

1. ***Certainly Factor***

Teori Certainty Factor (CF) diusulkan oleh Shorliffe dan Buchanan pada 1975 untuk mengakomodasikan ketidakpastian pemikiran (*inexact reasoning*) seorang pakar. Untuk mengakomodasi hal ini digunakan untuk menggambarkan tingkat keyakinan pakar terhadap masalah yang sedang dihadapi . Dalam mengekspresikan derajat keyakinan, certainty theory menggunakan suatu nilai yang disebut certainty factor untuk mengasumsikan derajat keyakinan seorang pakar terhadap suatu data. Konsep ini kemudian diformulasikan dalam rumus dasar pada Persamaan sebagai berikut.

CF[h, e] = MB[h, e] - MD[h, e]

di mana:

* **CF[h, e]** adalah CF hipotesis h berdasarkan bukti e.
* **MB[h, e]** adalah ukuran kepercayaan terhadap hipotesis h jika diberikan bukti e.
* **MD[h, e]** adalah uk
* uran ketidakpercayaan terhadap hipotesis h jika diberikan bukti e.

**Penjelasan Rumus:**

* **MB[h, e]** mewakili keyakinan terhadap hipotesis h ketika bukti e hadir. Nilai MB berkisar antara 0 dan 1, di mana 0 menunjukkan ketidakpercayaan penuh dan 1 menunjukkan kepercayaan penuh.
* **MD[h, e]** mewakili ketidakpercayaan terhadap hipotesis h ketika bukti e hadir. Nilai MD berkisar antara 0 dan 1, di mana 0 menunjukkan kepercayaan penuh dan 1 menunjukkan ketidakpercayaan penuh.
* **CF[h, e]** adalah hasil akhir dari penggabungan MB dan MD. Nilai CF berkisar antara -1 dan 1, di mana:
  + **CF = 1** menunjukkan kepastian penuh bahwa hipotesis h benar.
  + **CF = 0** menunjukkan ketidakpastian penuh, tidak ada bukti yang mendukung atau menentang hipotesis.
  + **CF > 0** menunjukkan keyakinan bahwa hipotesis h benar.
  + **CF < 0** menunjukkan keyakinan bahwa hipotesis h salah.

1. ***Unified Modeling Language (UML*)**

*Unified Modeling Language* (UML) adalah bahasa pemodelan visual yang digunakan untuk memvisualisasikan, memahami, dan membangun sistem perangkat lunak . UML menyediakan seperangkat diagram dan elemen yang dapat digunakan untuk memodelkan berbagai aspek sistem perangkat lunak, termasuk struktur, perilaku, dan interaksi. UML dikembangkan oleh Object Management Group (OMG) dan telah menjadi standar industri untuk pemodelan sistem perangkat lunak.

1. Manfaat Penggunaan UML

Penggunaan UML menawarkan beberapa manfaat, antara lain:

* Meningkatkan komunikasi: UML menyediakan bahasa visual yang mudah dipahami oleh berbagai pemangku kepentingan, seperti pengembang perangkat lunak, analis bisnis, dan pengguna akhir. Hal ini dapat meningkatkan komunikasi dan kolaborasi dalam proyek pengembangan perangkat lunak [7].
* Memperjelas desain: UML membantu memperjelas desain sistem perangkat lunak dengan menyediakan representasi visual yang komprehensif. Hal ini dapat membantu mengidentifikasi potensi masalah desain dan mempermudah proses pengujian dan pemeliharaan perangkat lunak.
* Mendokumentasikan sistem: UML dapat digunakan untuk mendokumentasikan sistem perangkat lunak secara detail. Dokumentasi UML dapat digunakan sebagai referensi untuk pengembangan, pengujian, dan pemeliharaan perangkat lunak di masa depan.
* Membuat prototipe: UML dapat digunakan untuk membuat prototipe sistem perangkat lunak. Prototipe ini dapat digunakan untuk menguji desain sistem dan mendapatkan umpan balik dari pengguna sebelum pengembangan sistem yang lengkap.

1. Jenis-Jenis Diagram UML

UML menyediakan berbagai jenis diagram untuk memodelkan berbagai aspek sistem perangkat lunak. Berikut adalah beberapa jenis diagram UML yang umum digunakan:

* Diagram Kasus Penggunaan (*Use Case Diagram*): Digunakan untuk memodelkan fungsionalitas sistem dan interaksi antara sistem dengan penggunanya [8].
* Diagram Kelas (*Class Diagram*): Digunakan untuk memodelkan struktur sistem, termasuk kelas, atribut, metode, dan hubungan antar kelas [Booch, 1994].
* Diagram Aktivitas (*Activity Diagram*): Digunakan untuk memodelkan alur kerja sistem, termasuk aktivitas, aliran kontrol, dan titik keputusan.
* Diagram Sekuen (*Sequence Diagram*): Digunakan untuk memodelkan interaksi antar objek dalam sistem, termasuk pesan yang dikirim dan diterima antar objek .
* Diagram Keadaan (*State Machine Diagram*): Digunakan untuk memodelkan perilaku objek dalam sistem, termasuk keadaan, transisi, dan peristiwa yang memicu transisi.

1. Proses Pemodelan dengan UML

Proses pemodelan dengan UML umumnya terdiri dari beberapa langkah, antara lain:

1. Analisis Persyaratan: Mengidentifikasi kebutuhan dan fungsionalitas sistem.
2. Pemodelan Kasus Penggunaan: Membuat diagram kasus penggunaan untuk memvisualisasikan fungsionalitas sistem dan interaksi antara sistem dengan penggunanya.
3. Pemodelan Kelas: Membuat diagram kelas untuk memvisualisaskan struktur sistem, termasuk kelas, atribut, metode, dan hubungan antar kelas.
4. Pemodelan Perilaku: Membuat diagram aktivitas, diagram sekuen, dan diagram keadaan untuk memvisualisasikan perilaku sistem, termasuk alur kerja, interaksi antar objek, dan perilaku objek.
5. Implementasi: Menerjemahkan model UML ke dalam kode program.
6. Pengujian: Menguji sistem perangkat lunak untuk memastikan bahwa sistem memenuhi persyaratan dan berfungsi dengan baik.
7. Penjelasan Jenis-Jenis UML
8. ***Use Case* *Diagram***

*Use Case diagram* merupakan pemodelan untuk tingkah laku sistem informasi yang dibuat. *Use Case diagram* bertujuan untuk menjelaskan perilaku sistem serta aktor-aktor yang akan berhubungan pada sekumpulan proses yang ada di sistem yang dibuat. Pada dasarnya *Use Case* digunakan untuk mengetahuai fungsi apa saja yang ada didalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi itu. Syarat penamaan *Use Case* didefinisikan sesederhana mungkin dan mudah dipahami. Simbol-simbol yang digunakan pada *Use Case diagram,* dapat dilihat pada tabel 2.2.

Table 2. 2 simbol-simbol use case diagram

| No | Gambar | Nama | | Keterangan |
| --- | --- | --- | --- | --- |
| 1 |  | *Actor* | | Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case*. |
| 2 |  | *Dependency* | | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*). |
| 3 |  | *Generalization* | | Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*). |
| 4 |  | *Include* | | Menspesifikasikan bahwa *use case* sumber secara *eksplisit*. |
| 5 |  | *Extend* | | Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan. |
| 6 |  | *Association* | | Apa yang menghubungkan antara objek satu dengan objek lainnya. |
| 7 |  | *System* | | Menspesifikasikan paket yang menampilkan sistem secara terbatas. |
| 8 |  | *Use Case* | | Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor |
| 9 |  | *Collaboration* | Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi). | |
| 10 |  | *Note* | Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi | |

1. ***Class Diagram***

Class *Diagram* merupakan diagram yang menunjukan *class*-*class* yang ada dari sebuah sistem dan hubungannya secara logika. *Class* *diagram* menggambarkan struktur statis dari sebuah sistem. Karena itu *class* *diagram* merupakan kekuatan dasar dari hampir setiap metode berorientasi objek termasuk *UML*. Simbol-simbol yang digunakan pada *Class Diagram,* dapat dilihat pada Tabel 2.3.

Table 2. 3 Simbol-simbol class diagram

| **No** | **Gambar** | **Nama** | **Keterangan** |
| --- | --- | --- | --- |
| 1 |  | *Generalization* | Hubungan dimana objek anak *(descendent)* berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*). |
| 2 |  | *Nary Association* | Upaya untuk menghindari asosiasi dengan lebih dari 2 objek. |
| 3 |  | *Class* | Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama. |
| 4 |  | *Collaboration* | Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor |
| 5 |  | *Realization* | Operasi yang benar-benar dilakukan oleh suatu objek. |
| 6 |  | *Dependency* | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri |
| 7 |  | *Association* | Apa yang menghubungkan antara objek satu dengan objek lainnya. |

1. ***Activity Diagram***

*Activity diagram* atau diagram aktivitas adalah diagram yang menggambarkan konsep aliran data/kontrol, aksi terstruktur serta dirancang dengan baik dalam suatu sistem [9]. Diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan actor. Simbol-simbol yang terdapat pada *activity diagram,* disajikan pada tabel 2.4.

Table 2. 4 Simbol-simbol activity diagram

| **No** | **Gambar** | **Nama** | **Keterangan** |
| --- | --- | --- | --- |
| 1 |  | *Activity* | Memperlihatkan bagaimana masing-masing *class* saling berinteraksi satu sama lain. |
| 2 |  | *Action* | *State* dari sistem yang mencerminkan eksekusi dari suatu aksi. |
| 3 |  | *Initial Node* | Bagaimana objek dibentuk atau diawali. |
| 4 |  | *Actifity Final Node* | Bagaimana objek dibentuk dan diakhiri. |
| 5 |  | *Fork Node* | Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran. |

1. *Tools* Pembangunan

Adapun alat yang digunakan dalam proses pembangunan sistem informasi manajemen keuangan ini yaitu.

1. *Visual Studio Code*

*Visual Studio Code* merupakan aplikasi *editor* *code* *open source* yang dikembangkan oleh *Microsoft* untuk sistem operasi *Windows*, *Linux*, dan *MacOS*. *Visual Studio Code* memudahkan dalam penulisan *code* yang mendukung beberapa jenis pemrograman, seperti *C++, C#, Java, Python, PHP, GO*. *Visual Studio Code* yang memiliki kemampuan untuk mengidentifikasi jenis bahasa pemrograman yang digunakan dan memberi variasi warna sesuai dengan fungsi dalam rangkaian *code*. *Visual Studio Code* juga telah terintegrasi ke *Github* yang memudahkan dalam proses *development*. Selain itu fitur lainnya adalah kemampuan untuk menambah *ekstensi* dimana para *developer* dapat menambah *ekstensi* untuk menambah fitur yang tidak ada di *Visual Studio Code* [10]. *Code editor* ini secara langsung mendukung bahasa pemrograman dengan bantuan *plugin* yang dapat dipasang melalui *marketplace* *Visual Studio Code* (*seperti C++, C#, Python, Go, Java*, dst). Jadi *visual studio code* salah satu jenis *code* *editor* yang bisa membantu dalam *development* penulisan program *aplikasi*.

1. *XAMPP*

*Xampp* merupakan paket *PHP* berbasis *Open Source*. *Xampp* mengkombinasi beberapa paket perangkat lunak berbeda ke dalam satu paket. Paket yang disediakan antara lain adalah *Apache*, *MySQL*, *PHP*, *FilZilla* *FTP* *Server*, *phpMyAdmin*, dan lain-lain [11]. *Xampp* digunakan dalam proyek untuk *database* [12]. Server *Xampp* mencoba memberikan basis data untuk penyimpanan data besar dengan bantuan *MySQL*, *PHP* dan *Perl* [13]. Jadi pada dasarnya *Xampp* merupakan *software* yang menyediakan paket untuk kebutuhan *development* program baik server maupun database.

1. *PHP*

*PHP* kepanjangan dari *Hypertext Preprocessor* merupakan bahasa pemrograman yang dibuat oleh Rasmus Lerdorf pada tahun 1994 dan dapat bekerja bersama ke dalam *HTML* maupun *javascript* [14]. *PHP* juga bahasa pemrograman *web* atau *scripting language* yang dijalankan di server, sekumpulan *script* yang digunakan untuk mengolah data *form* dari *web* [15]. *PHP* difungsikan untuk membangun suatu *website* dinamis yang menyatu dengan kode *HTML*, *HTML* digunakan sebagai pembangun atau pondasi dari kerangka *layout* *web* sedangkan *PHP* difungsikan sebagai prosesnya. Jadi pada dasarnya *PHP* merupakan bahasa pemrograman *server site* dimana *PHP* memiliki kemampuan dalam melakukan pemrosesan basis data agar *website* dapat menjadi dinamis dengan penggabungan struktur bahasa *server site* dan *database*.

1. *Framework* *Codeigniter* 4

*Framework* dalam bahasa indonesia dapat diartikan sebagai kerangka kerja atau kumpulan instruksi yang dikumpulkan dalam *Class* dan *Function-Function* dengan fungsi masing-masing untuk memudahkan *developer* dalam memanggilnya tanpa harus menulis *syntax* program yang sama berulang-ulang serta dapat menghemat waktu [16]. *Framework* juga untuk mengembangkan aplikasi berbasis *website* maupun *desktop*. Kerangka kerja sangat membantu *developer* dalam menuliskan sebuah dengan lebih terstruktur dan tersusun rapi. Sejak *Codeigniter 4* pertama kali dirilis pada 2 Februari 2020 bertepatan dengan tanggal lahir Jim Parry sebagai *project leade*r *codeigniter* yang meninggal pada tanggal 15 Januari 2020, *codeigniter* versi 4 memberikan banyak perubahan yang sangat signifikan dengan versi sebelumnya yaitu *codeigniter* versi 3 [17]. *Framework Codeigniter versi 4* dapat mempercepat pekerjaan karena telah memiliki banyak dukungan *library* yang bisa digunakan seperti *autentikasi, filtering, modeling, controlling*. *Framework Codeigniter 4* mendukung modul *entitas* yang berguna untuk pemanggilan dan menghubungkan *entitas* lainnya tanpa harus membuat *code* baru.

1. *Mysql*

*My Structure Query Language* (*MySQL*) yaitu sebuah *software database*, yang merupakan tipe data relasional yang artinya *MySQL* penyimpanan datanya dalam bentuk tabel-tabel yang saling berhubungan [18].*MySQL* adalah sistem manajemen *database relasional (RDBMS)* *open-source* berbasis *SQL* yang bekerja dengan model *client-server*. Sistem *database MySQL* mendukung fitur seperti *multithreaded*, *multi-user* dan *SQL Database Manajemen Sistem* (*DBMS*) [19]. Kegunaan atau fungsi *MySQL* adalah untuk data *warehousing* (gudang data), yaitu pengumpulan data dari berbagai sumber, untuk *e-commerce*, maupun aplikasi *logging*.

1. *Enterprise Architect*

*Enterprise Architect a*dalah pengorganisasian logis untuk proses bisnis dan infrastruktur teknologi informasi terkait dengan kebutuhan integrasi dan standarisasi dari sebuah *Operating* *Model* perusahaan. Elemen pokok pada arsitektur enterprise berbeda-beda, tergantung pada *Operating Model* yang digunakan [20]. *Enterprise Architect* digunakan untuk melakukan perancangan desain model yang dibutuhkan saat akan membangaun sistem atau aplikasi komputer.

1. *Figma*

*Figma* adalah aplikasi desain berbasis *cloud* dan alat *prototyping* untuk proyek digital [21]. *Figma* biasanya digunakan oleh desainer *UI/UX* untuk membangun sebuah proyek. *Figma* mampu berkolaborasi pada projek tertentu sehingga aplikasi ini sangat membantu dalam bekerja secara tim. *Figma* dapat dijalankan pada *browser* sehingga perangkat apapun dapat membuka aplikasi ini melalui *browser* namun dianjurkan menggunakan dalam mode *desktop* atau menggunakan perangkat yang memiliki ukuran layar lebar agar lebih nyaman dalam bekerja.

1. *Github*

*GitHub* adalah [layanan](https://id.wikipedia.org/wiki/Layanan_hos_web" \o "Layanan hos web) *[web](https://id.wikipedia.org/wiki/Layanan_hos_web" \o "Layanan hos web)* untuk proyek pengembangan [perangkat lunak](https://id.wikipedia.org/wiki/Perangkat_lunak" \o "Perangkat lunak) yang menggunakan sistem [kendali versi](https://id.wikipedia.org/wiki/Kendali_versi" \o "Kendali versi) [*Git*](https://id.wikipedia.org/wiki/Git) dan [layanan](https://id.wikipedia.org/wiki/Layanan_hosting_internet" \o "Layanan hosting internet) *[hosting](https://id.wikipedia.org/wiki/Layanan_hosting_internet" \o "Layanan hosting internet)**[internet](https://id.wikipedia.org/wiki/Layanan_hosting_internet" \o "Layanan hosting internet)*. Hal ini banyak digunakan untuk kode komputer. *Git* memberikan kontrol akses dan beberapa fitur kolaborasi seperti pelacakan *bug*, permintaan fitur, manajemen tugas, dan [wiki](https://id.wikipedia.org/wiki/Wiki) untuk setiap proyek [22]. *Git* adalah sistem kontrol versi *open-source* khusus yang dibuat oleh Linus Torvalds pada tahun 2005. Secara khusus, *Git* adalah sistem kontrol versi terdistribusi, yang berarti bahwa seluruh basis kode dan riwayat tersedia disetiap komputer pengembang, yang memungkinkan percabangan dan penggabungan dengan mudah. Menurut survei pengembang *Stack Overflow*, lebih dari 87% pengembang menggunakan *Git* [23]. Jadi *github* adalah *platform* atau *tools* yang berguna untuk melakukan manajemen proyek untuk pengembangan suatu proyek baik untuk menyimpan ataupun melakukan manajemen kontrol.

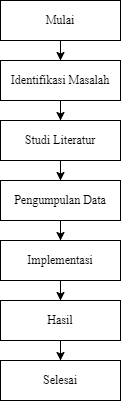
1. ***Blackbox testing***

*Blackbox testing* merupakan pengujian yang dilakukan agar tidak terjadi kesalahan alur program yang telah dibuat. Metode *Blackbox Testing* adalah sebuah metode yang dipakai untuk menguji sebuah *software* tanpa harus memperhatikan detail software [24]. Pengujian perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program untuk mengetahui apakah fungsi, masukan dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan . Jadi *blackbox* *testing* merupakan salah satu metode untuk melakukan pengujian sistem, apakah hasil *input* sistem sesuai dengan keinginan pengguna sistem.

# BAB III METODOLOGI PENELITIAN

## Tahap Penelitian

Dalam membangun sebuah sistem informasi dibutuhkan adanya persiapan dan perencanaan, dimana pada tahap tersebut memerlukan data-data untuk mendukung terlaksananya suatu sistem. Adapun tahapan penelitian ini agar dapat memberikan hasil yang optimal, maka dilakukan beberapa tahap seperti analisis, pengumpulan data, desain sistem, pengujian dan hasil. Tahap penelitian dapat dilihat pada gambar 3.1.



Gambar 3. 1 Tahap Penelitian

## Identifikasi Masalah

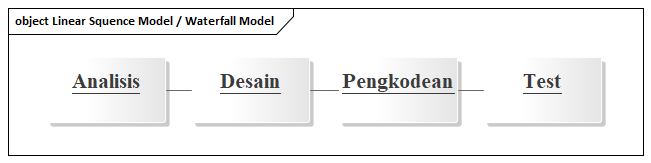
Tahap awal pada penelitian ini adalah identifikasi masalah. Permasalahan yang ada di penelitian ini yaitu ingin mengetahui tingkat akurasi metode certainly factor dalam melakukan diagnosa penyakit pada ayam petelur menggunakan metode *Certainty Factor*. Metode *Certainty Factor* digunakan untuk mengukur tingkat kepercayaan atau kepastian dalam proses diagnosa berdasarkan gejala-gejala yang muncul. Dalam penelitian ini, identifikasi masalah dilakukan dengan mengumpulkan data mengenai berbagai gejala yang dialami oleh ayam petelur yang sakit dan menghubungkannya dengan kemungkinan penyakit yang diderita. Hal ini penting untuk memastikan bahwa diagnosa yang diberikan memiliki tingkat akurasi yang tinggi sehingga dapat membantu dalam pengambilan keputusan yang tepat dalam penanganan dan pengobatan ayam petelur. Penelitian ini bertujuan untuk menguji seberapa efektif metode *Certainty Factor* dalam memberikan diagnosa yang akurat berdasarkan data gejala yang ada..

## Pengumpulan data

Pengumpulan data dalam penyusunan skripsi ini yaitu melalui Studi Kepustakaan, merupakan jenis metode literatur yang dilakukan dengan mengumpulkan data-data yang ada dengan topik permasalahan yang bersifat teoritis dengan cara membaca buku, makalah, jurnal, artikel, dan bahan kuliah. Pada penelitian ini data yang digunakan berasal dari jurnal “Sistem Pakar Diagnosa Penyakit Pada Ayam Dengan Metode *Certainty Factor* Berbasis Android”[3]. Data tersebut dianggap layak digunakan karena telah dipublikasikan pada media jurnal ter-Akreditasi.

## Metode Pengembangan Sistem

Metode yang digunakan dalam pengembangan sistem menggunakan metode *waterfall* yang merupakan model pembangunan sistem yang digunakan untuk merancang sistem sebuah sistem. *Waterfall* sering juga disebut sebagai metode *Linear Sequence Model* dimana proses pengembangan perangkat lunak sekuensial, kemajuan dilihat melalui fase analisis, desain, pengkodean, dan test/pengujian [25].



Gambar 3. 2 waterfall

Metode *waterfall* memiliki tahapan-tahapan sebagai berikut:

1. Analisis

Fase analisis merupakan fase untuk menentukan kebutuhan apa saja yang dibutuhkan dalam membangun sistem manajemen keuangan berdasarkan keinginan perusahaan.

1. Desain

Fase desain merupakan fase perancangan meliputi perancangan *Entity Relationship Diagram (ERD)* untuk menggambarkan *database* dari bentuk *entitas*, atribut, dan *relationship*. Desain Diagram Aktivitas, *Use Case Diagram*, *Sequence Diagram* dan juga melakukan perancangan halaman *userinterface*.

1. Pengkodean

Penulisan kode program menggunakan bahasa pemrograman *PHP framework codeigniter* versi 4 dan database *mysql*.

1. *Testing/* Pengujian

*Testing* atau pengujian merupakan proses untuk menemukan kesalahan pada sistem dengan menggunakan pengujian *blackbox*. Pada tahapan ini akan ditemukan mengenai akurasi fungsionalitasnya.

## Metode yang diusulkan

Pada penelitian ini terdapat 4 tahapan yang dilakukan, dapat dilihat pada Gambar 3.

### Analisis

Pada tahap analisis akan dilakukan analisis terhadap sistem pakar yang akan dibangun. Berikut ini adalah proses analisis yang dilakukan.

1. Akuisisi Pengetahuan

Akuisisi pengetahuan pakar didapat melalui jurnal pada penelitian sebelumnya [3]. Data penyakit ayam dapat dilihat pada Tabel 3.1.

Tabel 3. 1 Data Penyakit Ayam

|  |  |
| --- | --- |
| **Id Penyakit** | **Nama Penyakit** |
| **P01** | Gumboro (*Gumboro Disease*) |
| **P02** | Mareks |
| **P03** | Produksi Telur |
| **P04** | Tipus Ayam |
| **P05** | Berak Darah (*Coccidosis*) |
| **P06** | Salesma Ayam |

Gejala Penyakit pada ayam dapat dilihat pada table 3.2.

Tabel 3. 2 Gejala Penyakit pada ayam

|  |  |
| --- | --- |
| **Kode** | **Nama Gejala** |
| **G01** | Nafsu makan berkurang |
| **G02** | Tampak lesu |
| **G03** | Mencret keputih-putihan |
| **G04** | Tidur paruhnya diletakkan di lantai |
| **G05** | Duduk dengan sikap membungkuk |
| **G06** | Napas cepat |
| **G07** | Muka pucat |
| **G08** | Sempoyongan |
| **G09** | Kaki pincang |
| **G10** | Sayap menggantung |
| **G11** | Produksi telur menurun |
| **G12** | Kualitas telur jelek |
| **G13** | Mencret kehijau-hijauan |
| **G14** | Bulu kusam dan mengkerut |
| **G15** | Diare |
| **G16** | Kelihatan ngantuk dan bulu berdiri |
| **G17** | Badan Kurus |
| **G18** | Mencret bercampur darah |
| **G19** | Bersin-bersin |
| **G20** | Kelopak mata kemerahan |
| **G21** | Keluar nanah dari mata dan bau |
| **G22** | Pembengkakan dari sinus dan mata |

1. **Basis Pengetahuan**

Hal yang pertama kali dilakukan dalam membangun sistem pakar adalah membuat basis pengetahuan. Basis pengetahuan merupakan kumpulan fakta beserta aturan-aturannya. Basis pengetahuan dalam bentuk *if-then rule* penyakit pada ayam terlihat pada Tabel 3.3.

Tabel 3. 3 Rules

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Code*** | ***If*** | **Premis** | ***Then*** | **Konklusi** |
| **R1** | *If* | G1,G2,G3,G4,G5 | *Then* | P1 |
| **R2** | *If* | G6,G7,G8,G9,G10 | *Then* | P2 |
| **R3** | *If* | G11,G12,G13 | *Then* | P3 |
| **R4** | *If* | G14,G15,16 | *Then* | P4 |
| **R5** | *If* | G17,G18 | *Then* | P5 |
| **R6** | *If* | G19,G20,G21 | *Then* | P6 |

1. **Mekanisme *Inference Engine***

Untuk mendapatkan kesimpulan beserta nilai persentase kepastian dari beberapa fakta yang ada maka ditambahkan sebuah metode kepastian yaitu metode *Certainty Factor*. Berikut contoh perhitungan manualnya: Diketahui ayam mengalami gejala nafsu makan berkurang, ayam sesak nafas/megap-megap, badan kurus, bulu kusam dan berkerut, jengger pucat dan perut membesar.

CFpakar(Nafsu makan berkurang) = 0,23

CFpakar(Ayam sesak nafas/megap-megap) = 0,37

CFpakar(Badan kurus) = 0,43

CFpakar(Bulu kusam dan berkerut) = 0,40

CFpakar(Jengger pucat) = 0,43

CFpakar(Perut membesar) = 0,80

Kemudian kombinasikan masing-masing nilai CF tersebut.  
CFcombine CF[H,E]1,2 = CF[H,E]1 + CF[H,E]2 \*(1 – CF[H,E]1)  
= 0,23 + 0,37 \* (1 – 0,23) = 0,515old1

CFcombine CF[H,E]old1,3 = CF[H,E]old1 + CF[H,E]3 \*(1 - CF[H,E]old1)  
= 0,515 + 0,43\*(1-0,515) = 0,723old2

CFcombine CF[H,E]old2,4 = CF[H,E]old2 + CF[H,E]4 \*(1 - CF[H,E]old2)  
= 0,723 + 0,40\*(1-0,723) = 0,833old3

CFcombine CF[H,E]old3,5 = CF[H,E]old3 + CF[H,E]5 \* (1 - CF[H,E]old3)  
= 0,833 + 0,43\*(1-0,833) = 0,904old4

CFcombine CF[H,E]old4,6 = CF[H,E]old4 + CF[H,E]6 \*(1 - CF[H,E]old4)= 0,904 + 0,80\*(1-0,904)= 0,980old5

Persentase keyakinan = CF[H,E]old6 \*100%

= 0,980\*100% = 98%

Berdasarkan perhitungan di atas, maka ayam didiagnosa menderita penyakit Busung Ayam (*Lymphoid Leukosis*) dengan persentase keyakinan 98%.

### Desain

Fase desain merupakan fase perancangan meliputi perancangan *Entity Relationship Diagram (ERD)* untuk menggambarkan *database* dari bentuk *entitas*, atribut, dan *relationship*. Desain Diagram Aktivitas, *Use Case Diagram*, *Sequence Diagram* dan juga melakukan perancangan halaman *userinterface*.

### Pengkodean

Penulisan kode program menggunakan bahasa pemrograman *PHP framework codeigniter* versi 4 dan database *mysql*.

### Testing

*Testing* atau pengujian merupakan proses untuk menemukan kesalahan pada sistem dengan menggunakan pengujian *blackbox*. Pada tahapan ini akan ditemukan mengenai akurasi fungsionalitasnya.

# BAB IV HASIL DAN PEMBAHASAN

1. **Construction (Code & Test)**

Pada tahap ini, penelitian berfokus pada penulisan kode menggunakan bahasa PHP, diikuti dengan pengujian hasilnya menggunakan metode *blackbox testing*. Pengujian *blackbox* ini dilakukan untuk mengamati hasil input dan output dari perangkat lunak, guna memastikan apakah perangkat lunak berjalan sesuai dengan yang diharapkan atau tidak.

1. **Implementasi Perangkat *Hardware* dan *Software***

Pada tahap ini, peneliti membutuhkan spesifikasi perangkat keras dan lunak untuk mengoperasikan sistem pakar. Berikut adalah rinciannya:

Perangkat keras yang diperlukan untuk mengakses sistem pakar adalah sebagai berikut:

1. Prosesor: Intel Core I5 Gen-3
2. RAM: 8 GB
3. SSD: 256 GB

*Software* yang digunakan user untuk mengakses sistem pakar adalah browser chrome, edge, opra atau sejenisnya.

1. **Struktur Table**

Berikut ini adalah struktu tabel yang digunakan dalam membuat sistem :

Gambar 4. 1 Struktur Tabel Data Penyakit

|  |  |  |
| --- | --- | --- |
| No | Nama Atribut | Tipe |
| 1 | *Id* | *Int (Primary Key)* |
| 2 | *Code* | *String* |
| 3 | *Name* | *String* |
| 4 | *Description* | *String* |

Gambar 4. 2 Struktur Tabel Data Gejala

|  |  |  |
| --- | --- | --- |
| No | Nama Atribut | Tipe |
| 1 | *Id* | *Int (Primary Key)* |
| 2 | *Code* | *String* |
| 3 | *Name* | *String* |
| 4 | CfPakar | *String* |
| 5 | *Description* | *String* |

Gambar 4. 3 Struktur Tabel Data Rule

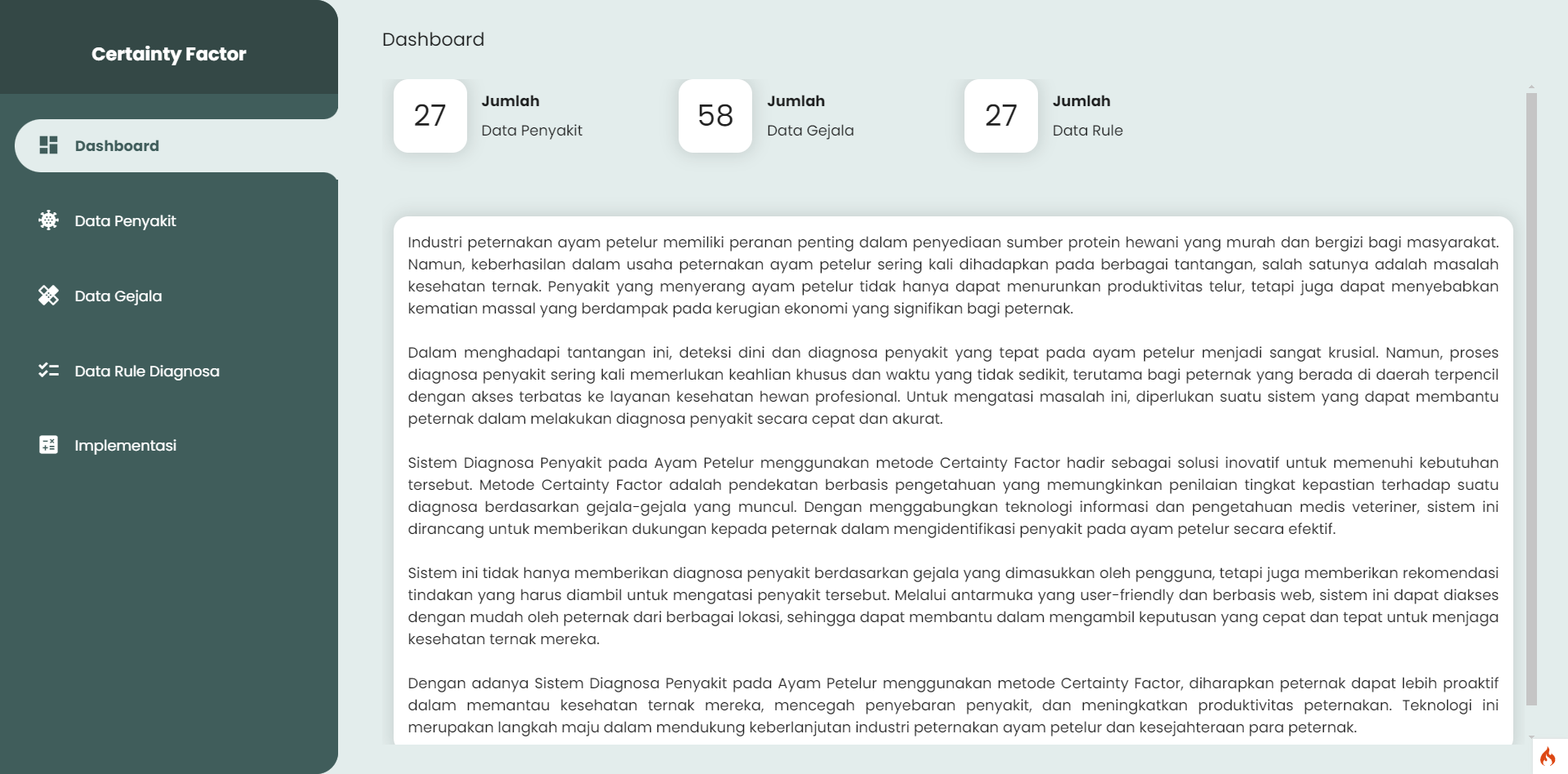
|  |  |  |
| --- | --- | --- |
| No | Nama Atribut | Tipe |
| 1 | *Id* | *Int (Primary Key)* |
| 2 | *Code* | *String* |
| 3 | Gejala\_id | *String* |
| 4 | *Effect*\_id | *String* |
| 5 | *Effect\_type* | *Enum(disease,symptom)* |

1. **Implementasi Antarmuka**

Berikut ini adalah antarmuka yang telah dibuat sebagai penghubung antara pengguna dengan sistem :

1. **Halaman *Dashboard***

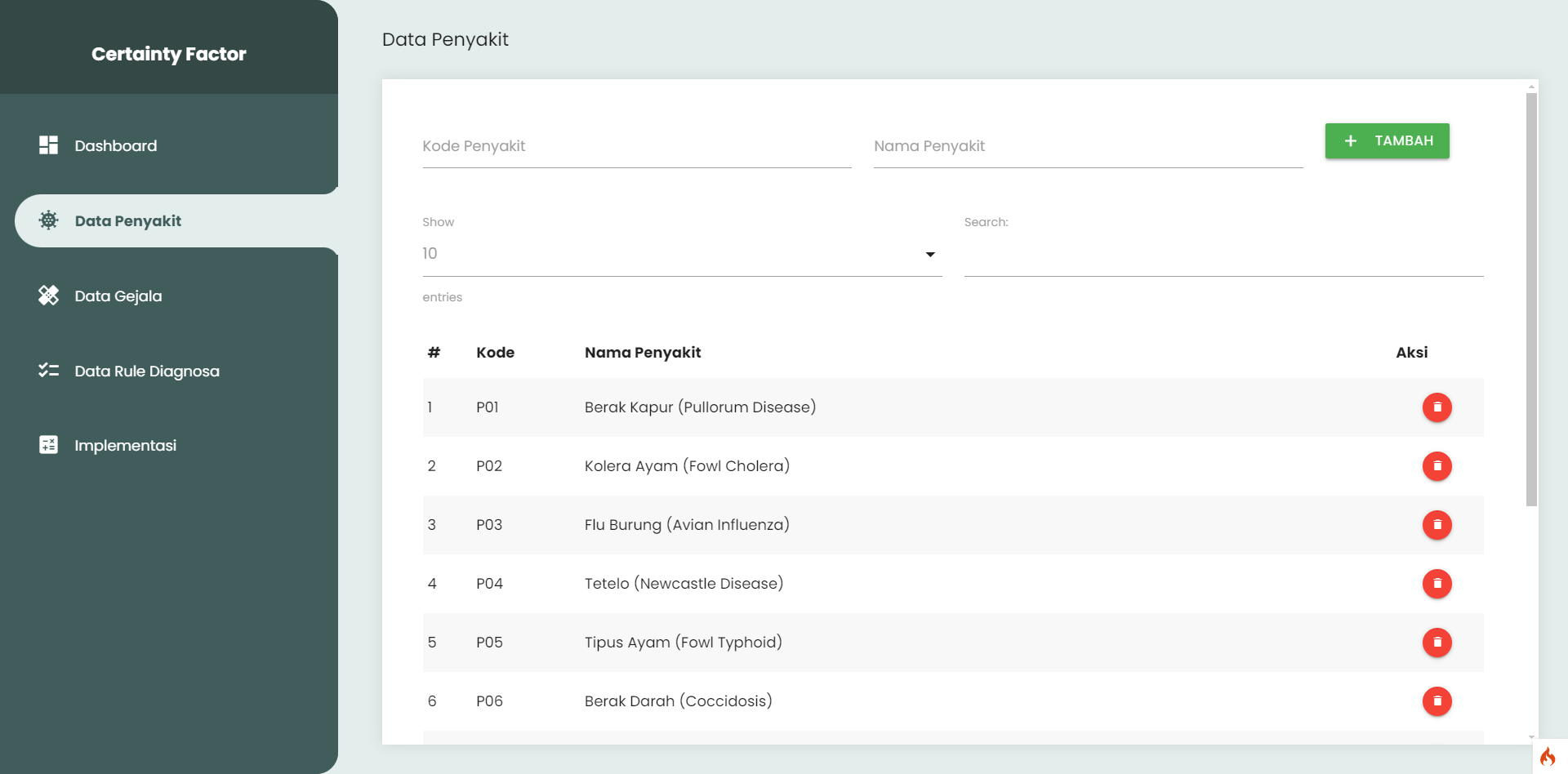
Halaman dashboard merupakan halaman utama yang akan di akses oleh pengguna dalam melakukan manajemen data-data utama sistem pakar.



Gambar 4. 5 Halaman Dashboard

1. **Halaman Data Penyakit**

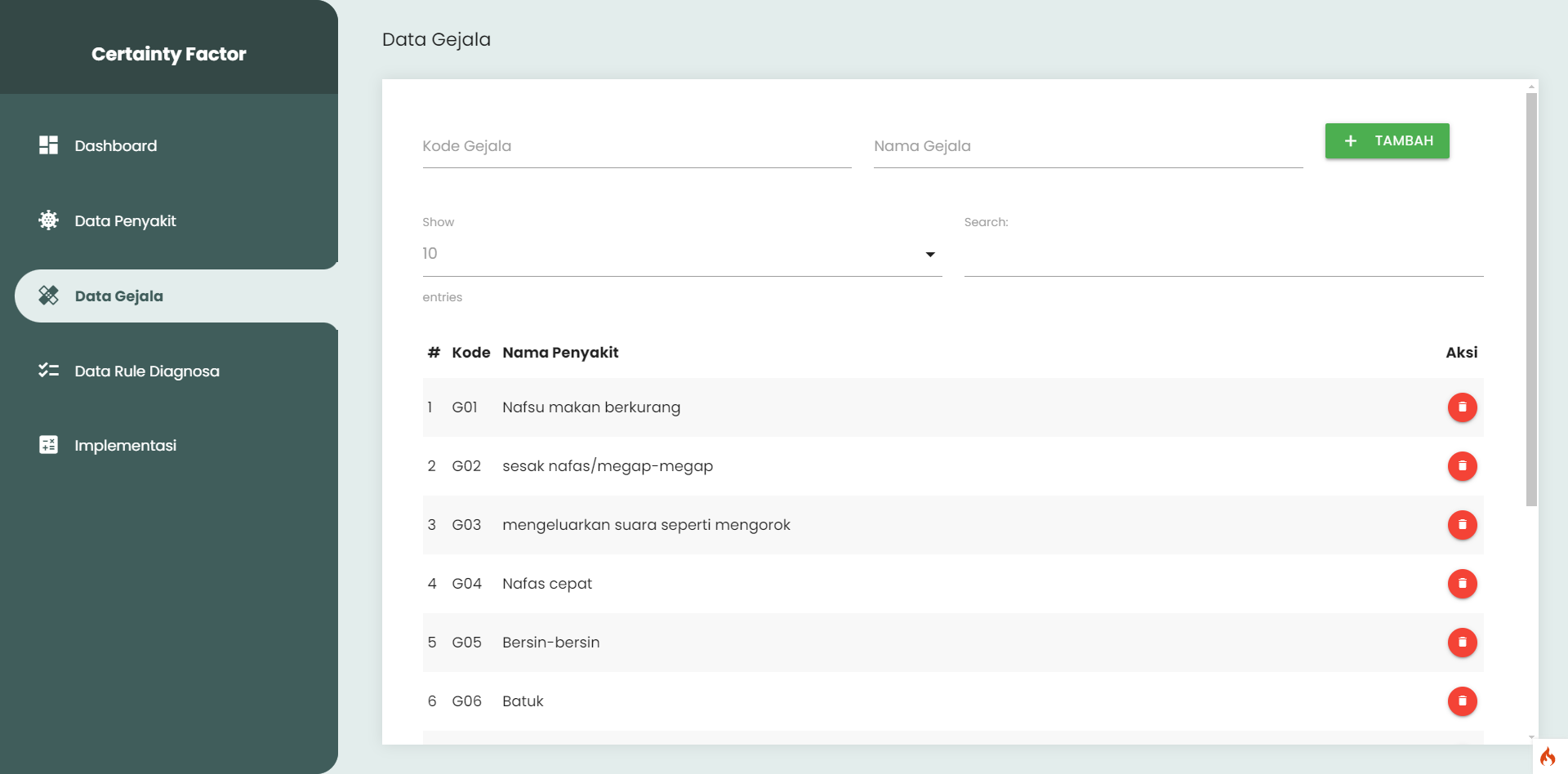
Halaman data penyakit merupakan halaman yang bergunakan sebagai userinterface dalam mengelola data penyakit.



Gambar 4. 6 Halaman Data Penyakit

1. **Halaman Data Gejala**

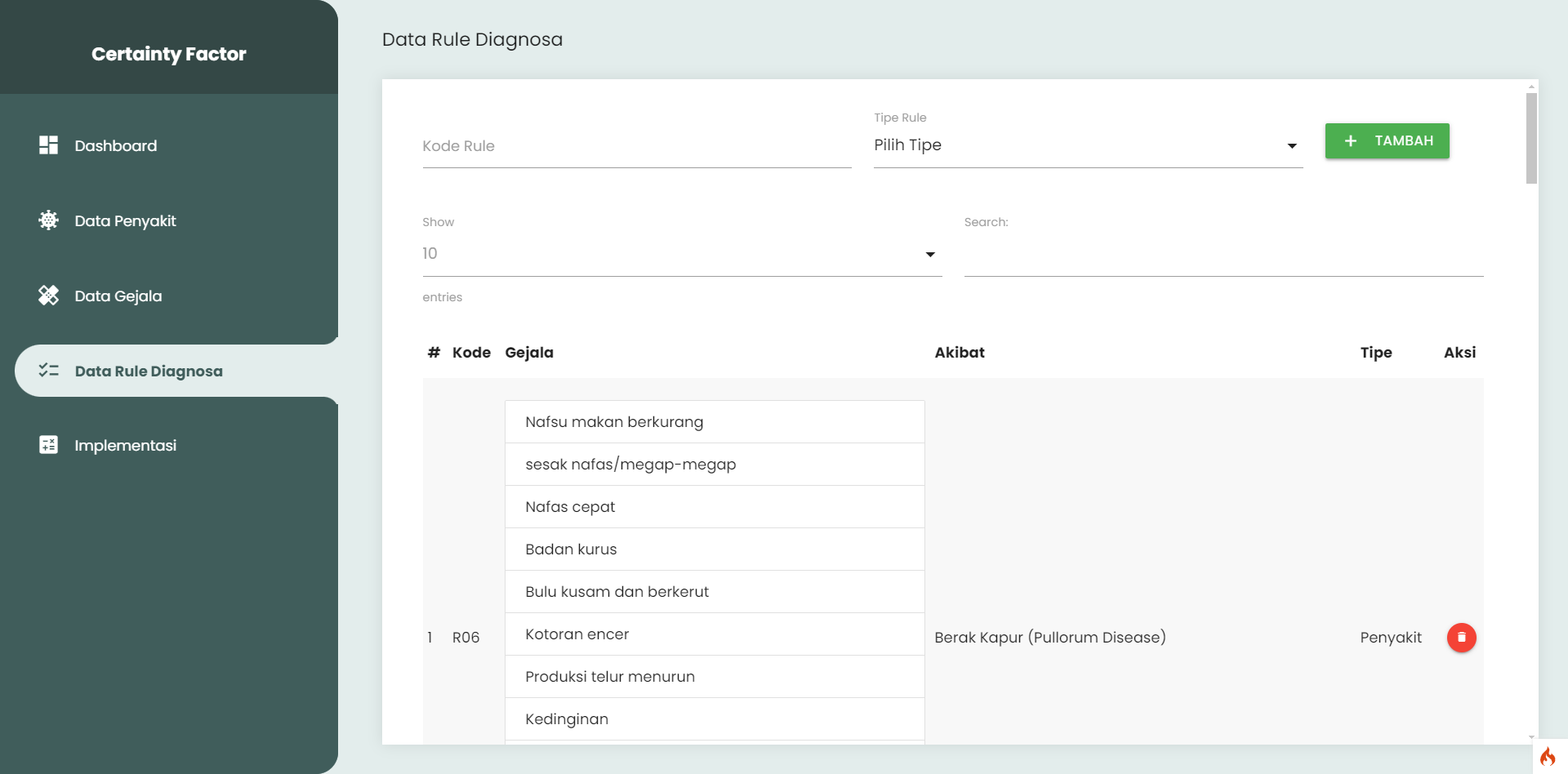
Halaman data gejala merupakan halam yang berfungsi sebagai antarmuka pengguna dalam mengelola data gejala.



Gambar 4. 7 Halaman Data Gejala

1. **Halaman Data Rule**

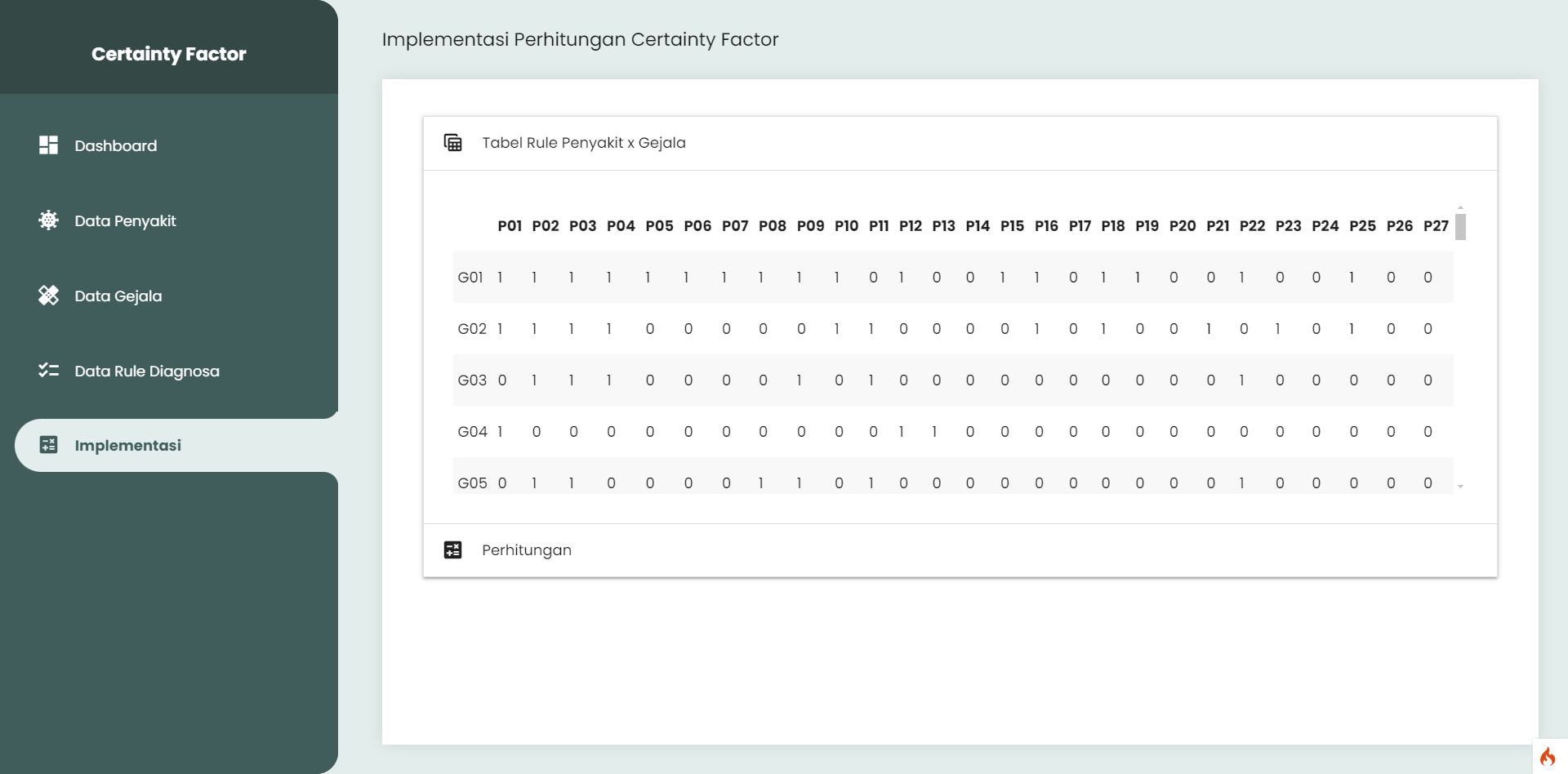
Halaman data rule merupakan antarmuka pengguna yang berfungsi untuk mengelola data rule sistem pakar.



Gambar 4. 8 Halaman Data Rule

1. **Halaman Implementasi**

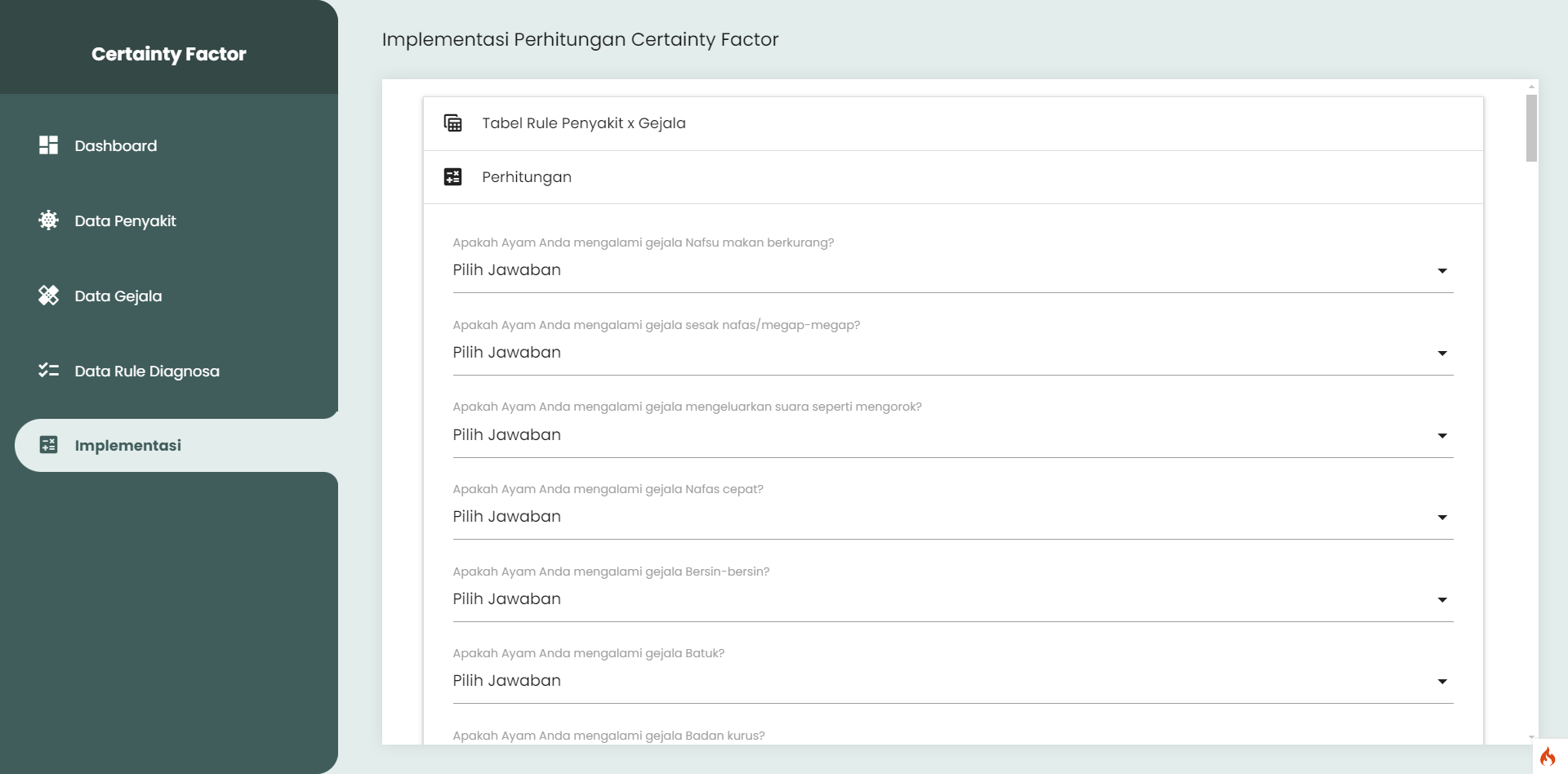
Halaman implementasi merupakan halaman yang berfungsi untuk melihat data utama perhitungan yang divisualisasikan menggunakan matrix.



Gambar 4. 9 Halaman Implementasi

1. **Halaman Perhitungan**

Halaman perhitungan merupakan antarmuka pengguna yang digunakan untuk melihat perhitungan menggunakan *certainly factor*.



Gambar 4. 10 Halaman Perhitungan

1. **Halaman Diagnosa**

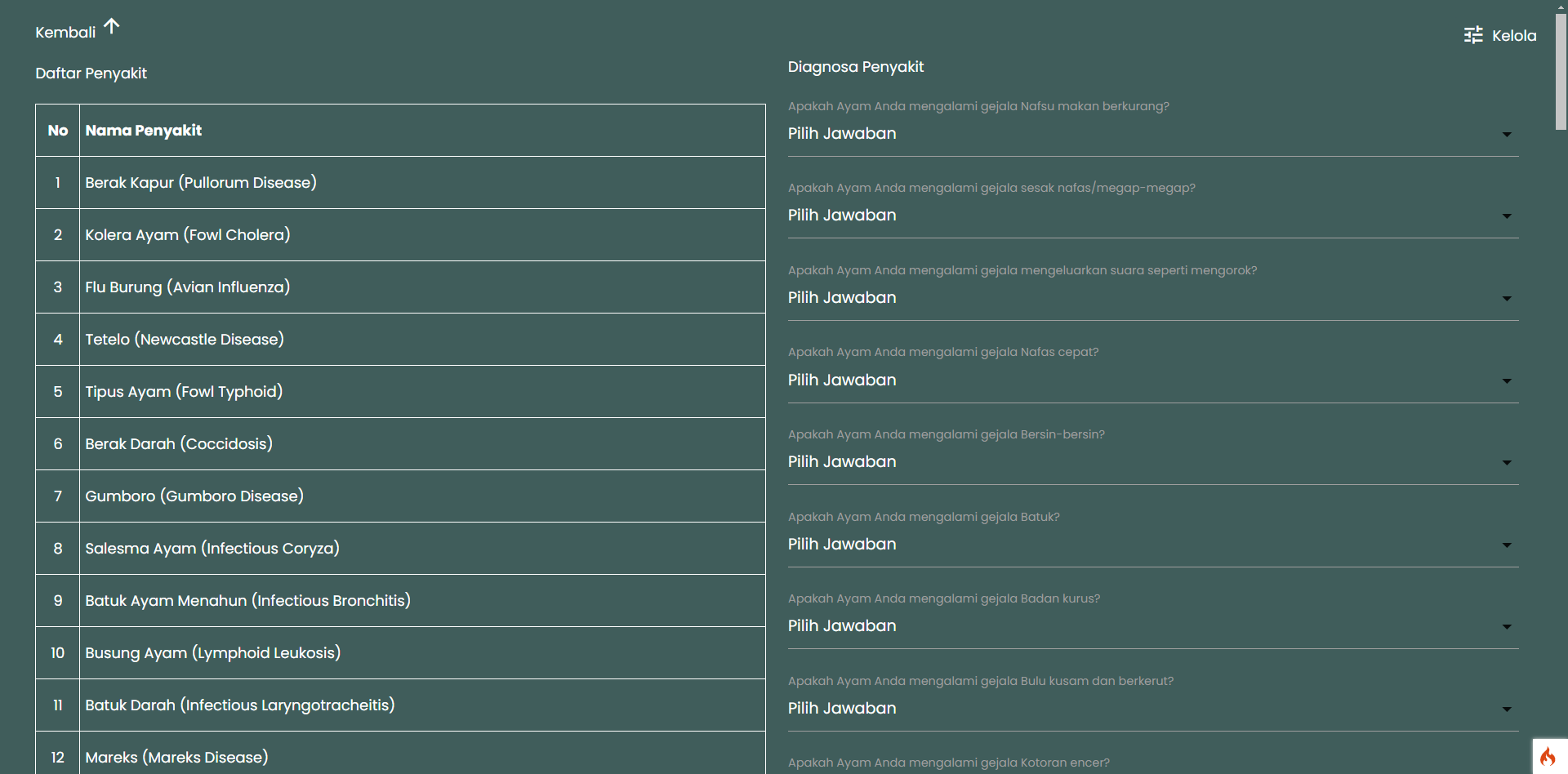
Halaman diagnose merupakan halaman utama atau landing page dari sistem pakar.



Gambar 4. 11 Halaman Diagnosa

1. **Halaman Proses Diagnosa**

Halaman proses diagnose merupakan halaman yang digunakan oleh user dalam melakukan input data-data yang dialami oleh hewan dan akan menghasilkan diagnosa sesuai dengan aturan yang telah dibangun.



Gambar 4. 12 Halaman Proses Diagnosa

1. **Pengujian**

Pengujian perangkat lunak atau bisa disebut *software testing* adalah proses pengujian program dengan tujuan mencari *error* atau kecacatan pada sebuah program sebelum dibagikan kepada user. Pada proses pengujian dilakukan menggunakan metode *blackbox* dan Uji Validitas.

1. **Uji *BlackBox***

Tabel 4. 1 Pengujian Kelola Data Penyakit

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Aksi** | **Output** | **Hasil** |
| 1 | Memilih menu data penyakit | Menu data penyakit ditampilkan | Berhasil |
| 2 | Menambah data penyakit dengan memasukan data penyakit seperti nama dll kemudian klik SIMPAN. | Menyimpan data yang sudah di masukan kedalam database | Berhasil |
| 3 | Mengedit data penyakit dengan memilih salah satu data penyakit kemudian klik edit dan akan mengarah ke formulir edit sesuai dengan data yang dipilih. | Mengedit data yang sudah diubah pada formulir | Berhasil |
| 5 | Menghapus data penyakit dengan memilih salah salah satu data penyakit. | Menghapus data yang dipilih | Berhasil |

Tabel 4. 2 Pengujian Kelola Data Gejala

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Aksi** | **Output** | **Hasil** |
| 1 | Memilih menu data gejala | Menu data gejala ditampilkan | Berhasil |
| 2 | Menambah data gejala dengan memasukan data penyakit seperti nama dll kemudian klik SIMPAN. | Menyimpan data yang sudah di masukan kedalam database | Berhasil |
| 3 | Mengedit data gejala dengan memilih salah satu data gejala kemudian klik edit dan akan mengarah ke formulir edit sesuai dengan data yang dipilih. | Mengedit data yang sudah diubah pada formulir | Berhasil |
| 5 | Menghapus data gejala dengan memilih salah salah satu data gejala. | Menghapus data yang dipilih | Berhasil |

Tabel 4. 3 Pengujian Kelola Data Rule

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Aksi** | **Output** | **Hasil** |
| 1 | Memilih menu data rule | Menu data rule ditampilkan | Berhasil |
| 2 | Menambah data rule dengan memasukan data rule kemudian klik SIMPAN. | Menyimpan data yang sudah di masukan kedalam database | Berhasil |
| 3 | Mengedit data rule dengan memilih salah satu data gejala kemudian klik edit dan akan mengarah ke formulir edit sesuai dengan data yang dipilih. | Mengedit data yang sudah diubah pada formulir | Berhasil |
| 5 | Menghapus data rule dengan memilih salah salah satu data rule. | Menghapus data yang dipilih | Berhasil |

Tabel 4. 4 Pengujian Perhitungan

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Aksi** | **Output** | **Hasil** |
| 1 | Memilih menu perhitungan | Menu perhitungan ditampilkan | Berhasil |
| 2 | Dapat melakukan perhitungan dengan mengirimkan data inputan gejala kemudian klik HITUNG. | Melakukan perhitungan dan menampilkan hasil perhitungan | Berhasil |

Tabel 4. 5 Pengujian Diagnosa

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Aksi** | **Output** | **Hasil** |
| 1 | Pengguna memasukan data-data gejala yang diderita kemudian setelah gejala dimasukan klik diagnosa. | Menampilkan hasil diagnosa | Berhasil |

1. **Uji Validitas**

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Rule** | **Aktual** | **Diagnosa** |
| 1 | RULE 1 | Gumboro (*Gumboro Disease*) | Gumboro (*Gumboro Disease*) |
| 2 | RULE 2 | Mareks | Mareks |
| 3 | RULE 3 | Produksi Telur | Produksi Telur |
| 4 | RULE 4 | Tipus Ayam | Tipus Ayam |
| 5 | RULE 5 | Berak Darah (*Coccidosis*) | Berak Darah (*Coccidosis*) |
| 6 | RULE 6 | Salesma Ayam | Salesma Ayam |

Berdasarkan Hasil Uji Validitas Diagnosa menghasilkan nilai yang cocok terhadap hasil diagnosa dari 6 rule dan 6 penyakit menghasilkan diagnose yang sesuai. Uji validitas diagnosa menghasilkan nilai akurasi sebesar 100%. Fungsi-fungsi yang ada pada aplikasi dapat berjalan sesuai dengan apa yang diharapkan yang diuji menggunakan metode *blackbox*.

# BAB V KESIMPULAN

1. **Kesimpulan**

Penelitian ini bertujuan untuk mengembangkan sistem diagnosis penyakit pada ayam petelur menggunakan metode *Certainty Factor (*CF). Berdasarkan hasil pengujian dengan uji *Blackbox*, sistem yang telah dikembangkan menunjukkan kemampuan yang baik dalam mendiagnosis penyakit pada ayam petelur dimana fungsi-fungsi yang ada didalamnya berjalan sesuai apa yang diharapkan. Hasil uji validitas diagnosa menghasilkan nilai akurasi sebesar 100% sehingga sistem pakar diagnosa penyakit pada ayam petelur ini dapat digunakan dengan dalam melakukan diagnosa.

1. **Saran**

Berdasarkan hasil penelitian ini, beberapa saran yang dapat diberikan untuk pengembangan lebih lanjut adalah sebagai berikut:

1. Peningkatan Basis Pengetahuan
2. Integrasi dengan Teknologi IoT
3. Pengembangan Fitur Tambahan

Dengan implementasi saran-saran di atas, diharapkan sistem diagnosa penyakit pada ayam petelur menggunakan metode Certainty Factor dapat lebih optimal dan memberikan manfaat yang lebih besar bagi peternak.

# DAFTAR PUSTAKA

[1] Kementerian pertanian, “Buku Statistik Peternakan dan Kesehatan Hewan Tahun 2022,” *Direktorat Jenderal Peternakan dan Kesehatan Hewan Kementerian Pertanian RI*, vol. 2, p. 90, 2023.

[2] Y. Toriqfai and S. Siswahyudianto, “Analisis Risiko Produksi Peternakan Ayam Petelur di Desa Sumberbendo Kecamatan Pucanglaban Kabupaten Tulungagung,” *Jurnal Pendidikan dan Kewirausahaan*, vol. 10, no. 2, pp. 614–634, 2022, doi: 10.47668/pkwu.v10i2.468.

[3] T. K. Ahsyar, T. D. Raharjo, and Syaifullah, “Sistem Pakar Diagnosa Penyakit Pada Ayam Dengan Metode Certainty Factor Berbasis Android,” *Jurnal Ilmiah Rekayasa dan Manajemen Sistem Informasi*, vol. 7, no. 2, pp. 166–172, 2021.

[4] J. Bere, J. Dedy Irawan, and F. Ariwibisono, “Sistem Pakar Diagnosis Penyakit Pada Ayam Menggunakan Metode Certainty Factor,” *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 5, no. 1, pp. 217–224, 2021, doi: 10.36040/jati.v5i1.3251.

[5] I. P. Sinaga, A. A. Soebroto, and I. Cholissodin, “Sistem Pakar untuk Diagnosis Penyakit Ayam menggunakan Metode Certainty Factor (Studi Kasus: Balai Besar Pelatihan Peternakan Batu),” *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 6, no. 6, pp. 2704–2714, 2022.

[6] F. Hamedan, A. Orooji, H. Sanadgol, and A. Sheikhtaheri, “Clinical decision support system to predict chronic kidney disease: A fuzzy expert system approach,” *International Journal of Medical Informatics*, vol. 138, p. 104134, Jun. 2020, doi: 10.1016/J.IJMEDINF.2020.104134.

[7] G. Booch, R. A. Maksimchuk, M. W. Engle, B. J. Young, J. Connallen, and K. A. Houston, *Object-oriented analysis and design with applications, third edition*, vol. 33, no. 5. 2008. doi: 10.1145/1402521.1413138.

[8] A. (Indiana U. Dennis, *System Analysis & Design with UML VERSION 2.0 AN OBJECT-ORIENTED APPROACH*. 2009.

[9] T. Arianti, A. Fa’izi, S. Adam, and ..., “Perancangan Sistem Informasi Perpustakaan Menggunakan Diagram Uml (Unified Modelling Language),” *Jurnal Ilmiah Komputer …*, vol. 1, no. 1, pp. 19–25, 2022.

[10] N. A. Ramdhan and D. A. Nufriana, “Rancang Bangun Dan Implementasi Sistem Informasi Skripsi Oline Berbasis WEB,” *Jurnal Ilmiah Intech : Information Technology Journal of UMUS*, vol. 1, no. 02, pp. 1–12, 2019, doi: 10.46772/intech.v1i02.75.

[11] R. Safitri, “Simple Crud Buku Tamu Perpustakaan Berbasis Php Dan Mysql :Langkah-Langkah Pembuatan,” *Tibanndaru : Jurnal Ilmu Perpustakaan dan Informasi*, vol. 2, no. 2, p. 40, 2018, doi: 10.30742/tb.v2i2.553.

[12] V. Nayak, P. R Nayak N, Sampoorna, Aishwarya, and N. H. Sowmya, “Agroxpert - Farmer assistant,” *Global Transitions Proceedings*, vol. 2, no. 2, pp. 506–512, 2021, doi: 10.1016/j.gltp.2021.08.016.

[13] R. Kannadasan, M. S. Saleembasha, and I. A. Emerson, “Survey on molecular cryptographic network DNA (MCND) using big data,” *Procedia Computer Science*, vol. 50, pp. 3–9, 2015, doi: 10.1016/j.procs.2015.04.003.

[14] R. K. Dewi, Q. J. Adrian, H. Sulistiani, and F. Isnaini, “Dashboard Interaktif Untuk Sistem Informasi Keuangan Pada Pondok Pesantren Mazroatul’ulum,” *Jurnal Teknologi dan Sistem Informasi (JTSI)*, vol. 2, no. 2, pp. 116–121, 2021.

[15] Fatimah and Samsudin, “Perancangan Sistem Informasi E-Jurnal Pada Prodi Sistem Informasi Diuniversitas Islam Indragiri,” *Jurnal Perangkat Lunak*, vol. 1, no. 1, pp. 33–49, 2019, doi: 10.32520/jupel.v1i1.782.

[16] M. Destiningrum and Q. J. Adrian, “Sistem Informasi Penjadwalan Dokter Berbassis Web Dengan Menggunakan Framework Codeigniter (Studi Kasus: Rumah Sakit Yukum Medical Centre),” *Jurnal Teknoinfo*, vol. 11, no. 2, p. 30, 2017, doi: 10.33365/jti.v11i2.24.

[17] J. Asmara, “Rancang Bangun Sistem Informasi Desa Berbasis Website (Studi Kasus Desa Netpala),” *Jurnal Pendidikan Teknologi Informasi (JUKANTI)*, vol. 2, no. 1, pp. 1–7, 2019.

[18] A. S. Bakhri and E. R. , Anggi Elanda, “Rancang Bangun Sistem Informasi Company Profile dan Monitoring Calon Pelamar Pada Perusahaan Berbasis Web (Studi Kasus : PT.Faros Bakti Utama Persada) Asep,” *Jurnal Interkom: Jurnal Publikasi Ilmiah Bidang Teknologi Informasi dan Komunikasi*, vol. 16, no. 3, pp. 24–31, 2021, doi: 10.35969/interkom.v16i3.175.

[19] Sudaria, A. S. Putra, and Y. Novembrianto, “Sistem Manajemen Pelayanan Pelanggan Menggunakan PHP Dan MySQL ( Studi Kasus pada Toko Surya ),” *Tekinfo*, vol. 22, no. 1, pp. 100–117, 2021.

[20] R. Yunis and T. Theodora, “Penerapan Enterprise Architecture Framework untuk Pemodelan Sistem Informasi,” *Jurnal SIFO Mikroskil*, vol. 13, no. 2, pp. 159–168, 2012, doi: 10.55601/jsm.v13i2.81.

[21] R. Pramudita, R. W. Arifin, A. N. Alfian, N. Safitri, and S. Dina, “Penggunaan Aplikasi Figma Dalam Membangun Ui / Ux Yang Interaktif Pada Program Studi Teknik,” vol. 3, no. 1, pp. 149–154, 2021.

[22] Wikipedia, “Github,” Wikipedia. [Online]. Available: https://id.wikipedia.org/wiki/GitHub

[23] C. HOSTING, “What Is GitHub? A Beginner’s Introduction to GitHub,” kinsta.com. [Online]. Available: https://kinsta.com/knowledgebase/what-is-github/

[24] F. C. Ningrum, D. Suherman, S. Aryanti, H. A. Prasetya, and A. Saifudin, “Pengujian Black Box pada Aplikasi Sistem Seleksi Sales Terbaik Menggunakan Teknik Equivalence Partitions,” *Jurnal Informatika Universitas Pamulang*, vol. 4, no. 4, p. 125, 2019, doi: 10.32493/informatika.v4i4.3782.

[25] V. Apriana and S. Fauziah, “Applying Waterfall Method on Sales Information System,” *Mantik*, vol. 5, no. 2, pp. 820–826, 2021.

# LAMPIRAN

**Lampiran 1. Source Code Perhitungan Certainly Factor**

public function implementasi()

    {

        $cfuser = collect([]);

        collect($this->**request**->getPost())->each(function ($value, $key) use (&$cfuser) {

            if ($value > 0) {

                $cfuser->push([

                    "code" => $key,

                    "value" => (double) $value,

                ]);

            };

        });

        $symptomps = SymptomModel::whereIn('code', $cfuser->pluck("code")->toArray())->get();

        $diseases = collect([]);

        $result = collect([]);

        RuleModel::whereIn("symptom\_id", $symptomps->pluck("id")->toArray())->each(function ($value, $key) use (&$diseases) {

            if (!in\_array($value->**disease**->**id**, $diseases->pluck("id")->toArray())) {

                $diseases->push($value->**disease**);

            }

        });

        $diseases->each(function ($value, $key) use (&$result, $cfuser) {

            $cfhe = collect([]);

            $value->**rules**->each(function ($rule, $key) use (&$cfhe, $result, $cfuser) {

                $cfe = $cfuser->where("code", $rule->**symptom**->**code**)->first();

                $cfhe->push((double) ($rule->**symptom**->**cfpakar** \* (isset($cfe['value']) ? $cfe['value'] : 0.0)));

            });

            $cfcombine = 0.0;

            $cfhe->each(function ($value, $key) use (&$cfcombine) {

                if ($key == 0) {

                    $cfcombine += $value;

                    return;

                }

                $cfcombine += $value \* (1 - $cfcombine);

            });

            $result->push([

                "penyakit" => $value,

                "cfhe" => $cfhe,

                "cfcombine" => $cfcombine,

            ]);

        });

        $result = $result->sortByDesc("cfcombine")->values();

        return $this->respond([

            "cfuser" => $cfuser,

            "result" => $result,

            "diseases" => $diseases,

        ]);

    }