International Journal Software Engineering and Computer Science (IJSECS)
6 (1), 2026, 15-24
Published Online April 2026 in IJSECS (http://www.journal.lembagakita.org/index.php/ijsecs)
P-ISSN: 2776-4869, E-ISSN: 2776-3242. DOI: https://doi.org/10.35870/ijsecs.v6i1.5400.

RESEARCH ARTICLE Open Access

Expert System for Identifying Hardware Damage Using Naïve Bayes Method (Case Study: Computer Laboratory of Sepuluh Nopember University Papua)

Isaac Samon Sabra [1*], **Heru Sutejo** [2], **Ajenkris Yanto Kungkung** [3]

[1*,2,3] Informatics Engineering Department, Universitas Sepuluh Nopember Papua, Jayapura City, Papua Province, Indonesia.

**Corresponding author*: jhosuasamonsabra@gmail.com

Received: August 20, 2025; Accepted: February 2, 2026; Published: April 1, 2026.

Abstract: This research aims to design and implement an expert system based on the Naïve Bayes method to identify computer hardware failures in the Computer Laboratory of Universitas Sepuluh Nopember Papua. The laboratory operates approximately 40 computer units used daily by students across multiple study programs, yet is supported by only three technicians — a gap that frequently delays repairs and disrupts practical sessions. The system draws on a knowledge base covering 15 hardware failure categories and 11 observable symptoms, including failures in processors, memory/RAM, storage devices (HDD/SSD), and peripheral components such as keyboards, mice, and monitors. Development followed the Waterfall model, system design was documented using UML, the application was built with CodeIgniter, and evaluation was conducted through accuracy testing against expert diagnoses. Testing on 20 cases yielded a 75% accuracy rate, demonstrating that the system is capable of supporting technicians in accelerating the troubleshooting process, reducing dependence on manual inspection, and sustaining the quality of laboratory practice sessions for students.

Keywords: Expert System; Naïve Bayes; Hardware Failure; Diagnosis; CodeIgniter.

1. Introduction

The rapid growth of computer usage has made computing devices central to nearly every domain of human activity — from professional work and academic tasks to daily communication. As dependency on these devices grows, so does the frequency of technical failures, particularly in environments where machines are used intensively and continuously (Waang Bler Tuang *et al.*, 2024; Perancangan *et al.*, 2024). In higher education settings, computers are not merely supplementary tools — they are the primary medium through which students conduct research, complete assignments, process data, and participate in practical sessions. Computer laboratories, in particular, carry a significant operational burden in study programs related to informatics, computer engineering, and information systems, functioning not only as training venues but as spaces where students test and apply theoretical knowledge directly (Sastypratiwi & Nyoto, 2021; Kalyzta & Syafrullah, 2023).

Universitas Sepuluh Nopember Papua operates a computer laboratory with approximately 40 units used almost daily across multiple study programs, yet is supported by only three technicians. At this level of utilization, hardware failures are not occasional anomalies — they are a recurring operational reality. When

diagnosis is slow and repairs are delayed, students are left without sufficient workstations, and the quality of practical instruction suffers. The problem is compounded by the nature of manual inspection: technicians must examine each machine individually, which is time-consuming and unsustainable when multiple failures occur simultaneously. A more structured diagnostic approach is therefore needed — one that can guide users through a symptom-based process and return a probable cause without requiring direct expert intervention at every step. Expert systems, a branch of artificial intelligence that encodes domain-specific knowledge to support decision-making, offer exactly this kind of solution (Saputra *et al.*, 2022; Hanafi *et al.*, 2023).

Several prior studies have addressed similar problems using different methods. Surya Pratama *et al.* (2022) developed a web-based application for detecting laptop and computer failures using Forward Chaining, where inference proceeds from observed symptoms toward a diagnosis — a method well-suited for rule-based, sequential reasoning. Arafiyah (2023) combined Naïve Bayes with Certainty Factor to diagnose laptop damage, achieving an 85% match rate against expert diagnoses across 40 test cases. Kurnia & Irfan (2024) applied Certainty Factor alone in a desktop application, producing a confidence percentage alongside each identified failure type. Each approach has its trade-offs: Forward Chaining is transparent but can become unwieldy as the rule base grows; Certainty Factor handles uncertainty well but requires careful calibration; the hybrid Naïve Bayes–Certainty Factor method improves diagnostic confidence but adds implementation complexity. The present study takes a more focused path — applying Naïve Bayes as a standalone classifier, specifically for hardware failure diagnosis in an educational laboratory context, using a knowledge base built from symptom data gathered directly from the field. The system was developed as a web application using CodeIgniter and tested against 20 real-world cases, yielding a 75% accuracy rate. This paper also presents a frank analysis of where the system fell short and what technical steps — including knowledge base expansion, symptom weighting, and potential hybrid method integration — could improve diagnostic reliability in future iterations.

2. Related Work

Research on expert systems for diagnosing computer and laptop hardware failures has explored a range of inference methods, each with distinct strengths and limitations. Surya Pratama *et al.* (2022) developed a web-based system using Forward Chaining, where inference proceeds from observed symptoms toward a conclusion — an approach effective for sequential, rule-driven diagnosis aimed at helping non-expert users identify hardware problems before consulting a technician. Saputra *et al.* (2022) extended this direction by combining Forward Chaining with Certainty Factor in a single web-based application, covering components such as processors, VGA cards, motherboards, memory, and hard disks, allowing the system to both trace symptom-based rules and quantify diagnostic confidence in one output. Kalyzta & Syafrullah (2023) applied Certainty Factor alongside a Decision Tree specifically within a university ICT laboratory setting, achieving 100% functional accuracy across 10 test cases under Black Box testing — a result that underscores the method's reliability in controlled, institutional environments. Kurnia & Irfan (2024) applied Certainty Factor alone in a desktop application, producing a confidence percentage alongside each diagnosis, which gave users a quantifiable measure of certainty without requiring direct expert involvement.

On the probabilistic side, Waang Bler Tuang *et al.* (2024) applied Naïve Bayes across ten hardware failure categories — including processor, memory, hard disk, and power supply — and reported differentiated probability scores for each class, demonstrating that the method can produce meaningful distinctions across multiple failure types. Hanafi *et al.* (2023) similarly proposed a web-based Naïve Bayes system for hardware fault detection, arguing that probabilistic classification is well-suited to this problem when historical symptom data is available. Arafiyah (2023) pushed this further by combining Naïve Bayes with Certainty Factor in a laptop diagnosis system built on CodeIgniter and MySQL, achieving an 85% match rate against expert diagnoses across 40 test cases — currently the highest reported accuracy among hybrid approaches in the reviewed literature. Aldisa (2022) took a different deployment path, implementing a Certainty Factor system as an Android application for laboratory use, covering four damage types across 12 symptom entries and evaluated through Alpha Testing with 20 participants, yielding a 54% suitability rating alongside full functional compliance in Black Box testing.

Across these studies, a consistent pattern emerges: most systems rely on Forward Chaining, Certainty Factor, or hybrid combinations, while Naïve Bayes — when used — is typically paired with a secondary method rather than applied independently. No study in this review applies Naïve Bayes as a standalone classifier within the specific context of an educational computer laboratory using symptom data gathered directly from field operations. The present study addresses that gap, evaluating the standalone performance of Naïve Bayes against 20 real-world diagnostic cases and providing a detailed analysis of where the method succeeds and where it requires further refinement.

3. Methodology

This study followed the Waterfall model for software development — a sequential, phase-by-phase approach first introduced by Winston Royce around 1970 and widely regarded as the most structured model in software engineering (Abdul Wahid, 2020). Also known as the Linear Sequential Model or classic life cycle, the Waterfall method is well-suited to projects with clearly defined requirements, as each phase must be completed before the next begins (Mailasari, 2023; Fitriyanto & Fitrani, 2024). The phases applied in this study are illustrated in Figure 1.

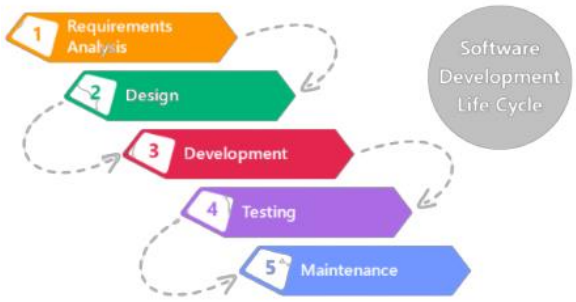

Figure 1. Waterfall Method Phases

3.1 Requirements Analysis

The requirements analysis phase aimed to define the scope and functional targets of the expert system. The system was designed to identify hardware failures in the Computer Laboratory of Universitas Sepuluh Nopember Papua, covering conditions such as computer failure to power on, Blue Screen of Death (BSOD), overheating, and related symptoms. At peak usage during practical sessions, the laboratory accommodates up to 31 students simultaneously, supported by only 3 technicians. The expected output of the system is a symptom-based diagnosis that identifies the most probable hardware failure and provides an initial handling recommendation. The knowledge base was constructed around 15 hardware failure categories (Table 1), 11 observable symptoms (Table 2), and 14 diagnostic rules (Table 3).

Table 1. Hardware Failure Types

Code	Component	Common Failure
K01	Power Supply (PSU)	No power output, voltage drop
K02	Motherboard	Component failure, short circuit
K03	RAM	Not detected, memory error
K04	HDD / SSD	Bad sector, unusual noise, not detected
K05	VGA / Display Card	Graphics chip failure, display artifacts
K06	Processor (CPU)	Overheating, complete shutdown
K07	Cooling Fan	Non-functional, fan failure
K08	Keyboard / Mouse	Port damage, connection lost
K09	Monitor	No signal received
K10	SATA Cable / Port	Loose connection, not detected
K11	Network Interface Card (NIC)	Unstable or absent connection
K12	USB Port	Not detected, port damage
K13	HDD Controller	Controller-level failure
K14	BIOS / Firmware	BIOS read error, corruption
K15	Chipset	Failure to detect certain hardware

Table 2. Symptom Codes

Code	Symptom
G01	Computer fails to power on
G02	Blank screen during boot
G03	Repeated beep sounds at startup
G04	Computer restarts spontaneously
G05	Blue Screen of Death (BSOD) during use
G06	Hard disk not detected
G07	Distorted or fragmented display
G08	Keyboard or mouse not detected

G09	Excessive heat
G10	Extremely slow system loading
G11	Unusual noise from hard disk or chassis

Table 3. Diagnostic Rules

No.	Code	Failure Name	Associated Symptoms
1	K01	Power Supply (PSU)	G01, G04, G11, G13, G15
2	K02	Motherboard	G01, G03, G04, G05, G15
3	K03	RAM	G03, G05, G10, G14, G15, G17
4	K04	HDD / SSD	G05, G06, G10, G11, G14, G15
5	K05	VGA / Display Card	G02, G07, G12, G17
6	K06	Processor (CPU)	G01, G04, G09, G10, G14, G15, G17
7	K07	Cooling Fan	G09, G10, G11
8	K08	Keyboard / Mouse	G08, G17
9	K09	Monitor	G01, G02, G07
10	K10	SATA Cable / Port	G06, G10, G13
11	K11	NIC	G16
12	K12	USB Port	G08, G13, G17
13	K13	HDD Controller	G03, G05, G06, G10, G11, G15
14	K14	BIOS / Firmware	G02, G03, G06, G15

3.2 System Design

System design was carried out using Unified Modeling Language (UML), which provides a standardized way to model system behavior and structure (Taufiq & Sandi, 2021). The design covers four diagram types: Use Case Diagram to define system functions and actors, Activity Diagram to map process flows, Sequence Diagram to trace interactions between system components, and Class Diagram to represent data structure (Damuri *et al.*, 2024). The user interface was prototyped using Balsamiq Wireframes. The overall system architecture is shown in Figure 2.

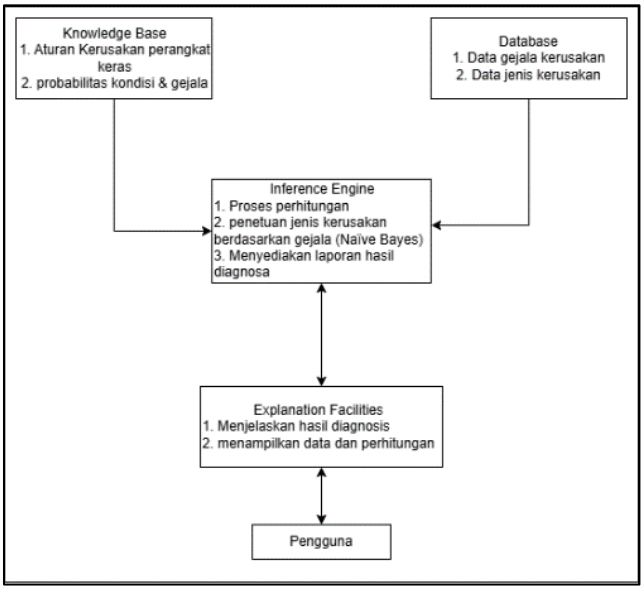

Figure 2. System Architecture

3.3 Implementation

The system was built using PHP as the server-side programming language (Widianto *et al.*, 2025), structured within the CodeIgniter framework (Anggraini *et al.*, 2020), and backed by a MySQL database (Hendriansyah, 2024). This technology stack was selected for its stability and compatibility with the web-based deployment model required for laboratory use.

3.4 Testing

System evaluation was conducted through accuracy testing, which compares system-generated diagnoses against diagnoses produced by a human expert (Riswandi Ishak *et al.*, 2020). This approach verifies that the system's logic and classification output align with real-world expert judgment across a defined set of test cases.

4. Result and Discussion

4.1 Results

The Naïve Bayes method is a technique for calculating the probability of an event based on the influence derived from historical data. Bayesian probability applies Bayes' theorem to handle uncertainty in data, utilizing prior experience to estimate the likelihood of an outcome (Referensi, 2020).

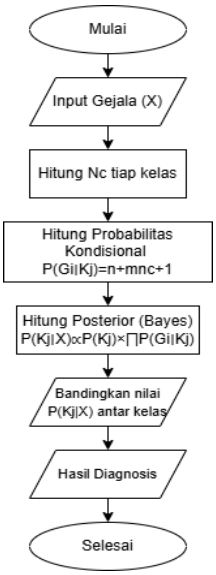

Figure 3. Flowchart Diagram

The flowchart above illustrates the calculation process of the Naïve Bayes method within the expert system. The process begins with symptom input (X) provided by the user. The system then calculates Nc, which represents the number of symptoms in each failure class. From this, the conditional probability of each symptom given a class is computed using the Laplace Smoothing formula $P(G_i \mid K_j) = \frac{n_c+1}{n+m}$. The system subsequently calculates the posterior probability using Bayes' theorem: $P(K_j \mid X) \propto P(K_j) \times \prod P(G_i \mid K_j)$. The posterior values of each class are then compared, and the class with the highest value is selected as the hardware failure diagnosis. The process concludes by displaying the diagnosis result to the user.

4.1.1 Naïve Bayes Reasoning Process

The following mathematical example illustrates the Naïve Bayes calculation for a case where the user reports symptoms G01, G04, G11, G13, and G15.

1) Calculating the nc value for each class

Number of symptoms per class (n) = 1, symptom weight per failure = 1/14 = 0.071, total symptoms (m) = 17.

Table 4. Failure K01 — Power Supply (PSU)

Symptom	Value
G01	1
G04	1
G11	1
G13	1
G15	1

Number of symptoms per class (n) = 1, symptom weight per failure = 1/14 = 0.071, total symptoms (m) = 17.

Table 5. Failure K02 — Motherboard

Symptom	Value
G01	1
G04	1
G11	0
G13	0
G15	1

Number of symptoms per class (n) = 1, symptom weight per failure = 1/14 = 0.071, total symptoms (m) = 17.

Table 6. Failure K03 — RAM

Symptom	Value
G01	0
G04	0
G11	0
G13	0
G15	1

And so on through failure K14.

2) Calculating P(ai | vj) and P(vj) for each class

a) Failure K01

$$P(G01 \mid K01) = \frac{1 + 17 + 0.071}{1 + 17} = 0.123$$
$$P(G04 \mid K01) = \frac{1 + 17 + 0.071}{1 + 17} = 0.123$$
$$P(G11 \mid K01) = \frac{1 + 17 + 0.071}{1 + 17} = 0.123$$
$$P(G13 \mid K01) = \frac{1 + 17 + 0.071}{1 + 17} = 0.123$$
$$P(G15 \mid K01) = \frac{1 + 17 + 0.071}{1 + 17} = 0.123$$

b) Failure K02

$$P(G01 \mid K02) = \frac{1 + 17 + 0.071}{1 + 17} = 0.123$$
$$P(G04 \mid K02) = \frac{1 + 17 + 0.071}{1 + 17} = 0.123$$
$$P(G11 \mid K02) = \frac{0 + 17 + 0.071}{1 + 17} = 0.067$$
$$P(G13 \mid K02) = \frac{0 + 17 + 0.071}{1 + 17} = 0.067$$
$$P(G15 \mid K02) = \frac{1 + 17 + 0.071}{1 + 17} = 0.123$$

b) Failure K03

$$P(G01 \mid K03) = \frac{0 + 17 + 0.071}{1 + 17} = 0.067$$
$$P(G04 \mid K03) = \frac{0 + 17 + 0.071}{1 + 17} = 0.067$$
$$P(G11 \mid K03) = \frac{0 + 17 + 0.071}{1 + 17} = 0.067$$
$$P(G13 \mid K03) = \frac{0 + 17 + 0.071}{1 + 17} = 0.067$$
$$P(G15 \mid K03) = \frac{1 + 17 + 0.071}{1 + 17} = 0.123$$

And so on through failure K14.

3) Calculating P(ai | vj) × P(vj) for each class

a) Failure K01

$$P(K01) \times [P(G01 \mid K01) \times P(G04 \mid K01) \times P(G11 \mid K01) \times P(G13 \mid K01) \times P(G15 \mid K01)]$$
$$= 0.1230 \times 0.1230 \times 0.1230 \times 0.1230 \times 0.1230 = 0.0000281712271385$$

b) Failure K02

$$P(K02) \times [P(G01 \mid K02) \times P(G04 \mid K02) \times P(G11 \mid K02) \times P(G13 \mid K02) \times P(G15 \mid K02)]$$
$$= 0.1230 \times 0.1230 \times 0.0675 \times 0.0675 \times 0.1230 = 0.0000084718882862$$

c) Failure K03

$$P(K03) \times [P(G01 \mid K03) \times P(G04 \mid K03) \times P(G11 \mid K03) \times P(G13 \mid K03) \times P(G15 \mid K03)]$$
$$= 0.0675 \times 0.0675 \times 0.0675 \times 0.0675 \times 0.1230 = 0.0000025477374763$$

And so on through failure K14. The complete probability results for all failure classes are presented in Table 7.

Table 7. Hardware Failure Probability Calculation Results (Naïve Bayes Method)

Code	Description	Result
K01	Power Supply (PSU)	0.0000281712271385
K02	Motherboard	0.0000084718882862
K03	RAM	0.0000025477374763
K04	HDD / SSD	0.0000046458742215
K05	VGA / Display Card	0.0000013971463580
K06	Processor (CPU)	0.0000084718882862
K07	Cooling Fan	0.0000025477374763
K08	Keyboard / Mouse	0.0000013971463580
K09	Monitor	0.0000025477374763
K10	SATA Cable / Port	0.0000025477374763
K12	NIC	0.0000013971463580
K13	USB Port	0.0000025477374763
K14	HDD Controller	0.0000046458742215
K15	BIOS / Firmware	0.0000025477374763

Based on the symptoms reported — G01, G04, G11, G13, and G15 — the system diagnosed a Power Supply (PSU) failure, with a final probability value of 0.0000281712271385, the highest among all classes.

4.1.2 User Interface

The interface implementation phase is the process of realizing the system's interface design into a form that can be directly used by the user.

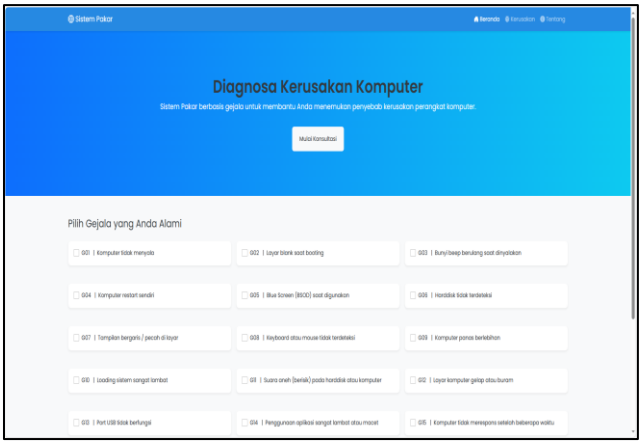

Figure 4. Computer Damage Diagnosis Page

The page shown in Figure 4 is the main display of the expert system for diagnosing computer hardware failures. Users can select various symptoms experienced by the device, such as the computer failing to power on, a blank screen, or unusual sounds from inside the computer. Once symptoms are selected, the system processes the input using the Naïve Bayes method to determine the most probable failure type. The interface is designed to be simple and interactive, featuring a "Start Consultation" button to allow users to perform a diagnosis quickly and conveniently.

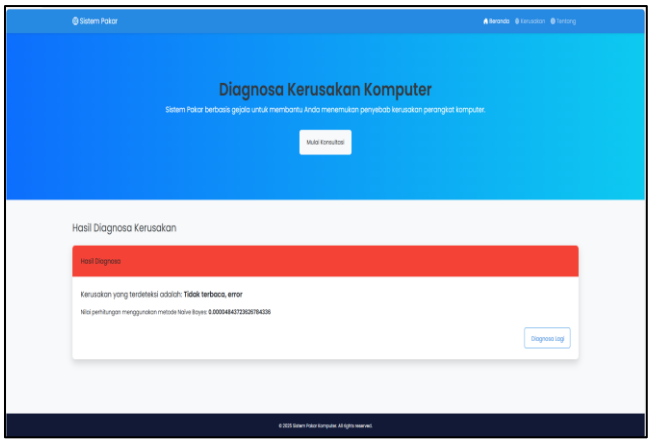

Figure 5. Diagnosis Result Page

The page shown in Figure 5 displays the diagnosis result after the user has selected the experienced symptoms. The expert system presents the detected failure type along with a description of the possible error condition and the probability value calculated using the Naïve Bayes method. This section provides the user with a final output regarding the most likely hardware failure, while also supporting technicians in determining the appropriate repair steps.

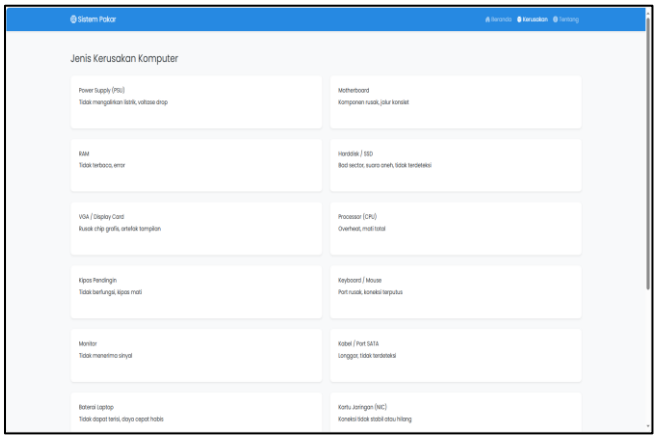

Figure 6. Failure Type List Page

The page shown in Figure 6 displays a list of computer failure types that can be identified by the expert system. Each failure type is accompanied by a brief description of the typical symptoms or conditions — for example, a Power Supply (PSU) failure is indicated by no power flow, while a RAM failure is indicated by the device not being detected. This information serves as the system's knowledge base as well as a reference for users to understand the causes of hardware failures. With a simple and structured presentation, this page helps both users and technicians recognize failure types before proceeding with repairs.

4.1.3 Accuracy Testing

To ensure diagnostic accuracy, the expert system was evaluated by comparing its output against diagnoses produced by a human expert. The test was conducted using 20 test cases.

Table 8. System Accuracy Testing Results

No.	Symptoms	System Diagnosis	Expert Diagnosis	Accuracy
1	G01, G04, G11, G13, G15	Power Supply (PSU)	Power Supply (PSU)	Accurate
2	G01, G03, G04, G05, G15	Motherboard	Motherboard	Accurate
3	G03, G05, G10, G14, G15, G17	RAM	RAM	Accurate
4	G05, G06, G10, G11, G14, G15	HDD / SSD	HDD / SSD	Accurate
5	G02, G07, G12, G17	VGA / Display Card	VGA / Display Card	Accurate
6	G01, G04, G09, G10, G14, G15, G17	Processor (CPU)	Processor (CPU)	Accurate
7	G09, G10, G11	Cooling Fan	Cooling Fan	Accurate
8	G08, G17	Keyboard / Mouse	Keyboard / Mouse	Accurate
9	G02, G07	VGA / Display Card	Monitor	Accurate
10	G06, G10, G13	SATA Cable / Port	SATA Cable / Port	Accurate

11	G16	NIC	NIC	Accurate
12	G08, G13, G17	USB Port	USB Port	Accurate
13	G03, G05, G06, G10, G11, G15	HDD Controller	HDD Controller	Accurate
14	G02, G03, G06, G15	BIOS / Firmware	BIOS / Firmware	Accurate
15	G03, G05, G09, G11	HDD Controller	RAM	Inaccurate
16	G07, G13, G14, G17	RAM	Motherboard	Inaccurate
17	G03, G05, G10, G15	RAM	RAM	Accurate
18	G08, G13, G14, G15, G17	RAM	USB Port	Inaccurate
19	G02, G03, G06, G09	BIOS / Firmware	Processor	Inaccurate
20	G04, G07, G10, G13	Power Supply	Power Supply	Accurate

4.2 Discussion

Out of 20 test cases, 15 were diagnosed accurately and 5 were inaccurate, yielding an overall accuracy rate of 75%. The inaccurate cases were primarily caused by the following factors. First, symptom overlap between failure classes — for example, RAM and HDD Controller share several symptoms, making it difficult for the classifier to distinguish between them. Second, a limited number of input symptoms in certain test cases was insufficient to provide a representative signal for the correct class. Third, an uneven distribution of symptom data across classes caused the system to favor certain failure categories. Fourth, the independence assumption inherent in the Naïve Bayes method does not fully reflect real-world conditions, where symptoms are often correlated. Fifth, some symptoms remain too general and ambiguous to serve as reliable discriminators between failure types. These factors collectively reduced the system's ability to differentiate between failures with similar symptom profiles, resulting in mismatches between system output and expert judgment in several cases.

5. Conclusion

This study successfully designed and implemented a Naïve Bayes-based expert system for diagnosing computer hardware failures in the Laboratory of Universitas Sepuluh Nopember Papua, utilizing a knowledge base containing symptom data and common failure types. Based on testing conducted on 20 case samples, the system produced accurate diagnoses in 15 cases (75%) and inaccurate results in 5 cases (25%), indicating a reasonably reliable performance level while still requiring further refinement. To improve accuracy in future development, several technical measures can be considered, including expanding the knowledge base with more specific symptom variations — such as repeated beep sounds during boot for detecting RAM or BIOS failures — using a larger and more diverse test dataset to achieve a more balanced symptom distribution, assigning weights to overlapping symptoms to reduce diagnostic ambiguity, and combining the Naïve Bayes method with other algorithms such as Decision Tree or Fuzzy Logic to address the limitations of the independence assumption between symptoms.

Acknowledgment

The authors would like to express their gratitude to all parties who have provided support in the preparation of this article entitled "Expert System for Identifying Hardware Failures Using the Naïve Bayes Method (Case Study: Computer Laboratory of Universitas Sepuluh Nopember Papua)." Special thanks are extended to the Computer Laboratory of Universitas Sepuluh Nopember Papua for their cooperation and the data provided, as well as to the supervising lecturer for the guidance, direction, and motivation offered throughout the writing process.

References

Aldisa, R. T. (2022). Penggunaan metode certainty factor pada sistem pakar deteksi kerusakan perangkat keras (hardware) komputer di laboratorium berbasis Android. *Journal of Information System Research (JOSH)*, *3*(3), 314–323. https://doi.org/10.47065/josh.v3i3.1528

Anggraini, Y., Pasha, D., Setiawan, A., & Setiawan, A. (2020). Sistem Informasi Penjualan Sepeda Berbasis Web Menggunakan Framework Codeigniter (Studi Kasus: Orbit Station). *Jurnal Teknologi Dan Sistem Informasi (JTSI)*, *1*(2), 64-70.

Arafiyah, R. (2023). Sistem Pakar Diagnosa Kerusakan Laptop Menggunakan Metode Naïve Bayes-Certainty Factor Berbasis Website. *J-KOMA: Jurnal Ilmu Komputer dan Aplikasi, 6*(2), 75–86.

Damuri, A., Malik, A., & Triyanto, I. (2024). Aplikasi Sistem Administrasi Surat Menyurat Berbasis Web Studi Kasus Desa Karang Bahagia Bekasi. *ALMUISY: Journal of Al Muslim Information System*, *3*(1), 57-64.

Fitriyanto, A., & Fitrani, A. S. (2024). Aplikasi penjualan tas di Indonesia berbasis web menggunakan metode waterfall. *Indonesian Journal of Applied Technology*, *1*(2), 32.

Hanafi, M. R., Simon, J., & Wahyuni, S. (2023). Analisis Perancangan Sistem Pakar Pendeteksi Kerusakan Hardware Pada Komputer Berbasis Web Dengan Metode Naive Bayes. *Warta Dharmawangsa*, *17*(3), 1190-1206. https://doi.org/10.46576/wdw.v17i3.3575.

Lisdarti, L., & Hendriansyah, R. (2024). Perancangan Sistem Informasi Surat Menyurat Berbasis Web (Studi Kasus: Dmptpsp Kabupaten Muaro Jambi). *Jurnal Akademika*, *16*(2), 26-32. https://doi.org/10.53564/akademika.v16i2.1181.

Kalyzta, J., & Syafrullah, M. (2023). Sistem pakar diagnosa kerusakan komputer dengan algoritma certainty factor pada Lab ICT Budi Luhur. *Skanika*, *6*(1), 12–21. https://doi.org/10.36080/skanika.v6i1.2996

Kurnia, & Irfan. (2024). Sistem Pakar Diagnosis Kerusakan Pada Komputer Menggunakan Metode Certainty Factor. *Journal Of Information System And Artificial Intelligence*, *4*(2), 98-104. https://doi.org/10.26486/jisai.v4i2.131.

Mailasari, M. (2023). Sistem informasi perpustakaan menggunakan metode waterfall. *Jurnal Sisfokom (Sistem Informasi dan Komputer)*, *8*(2), 207–214. https://doi.org/10.32736/sisfokom.v8i2.657

Ningki, C., & N. P. (2023). Implementasi aplikasi penjualan produk tradisional berbasis website menggunakan metode waterfall. *Informatika: Jurnal Ilmu Komputer*, *19*(2), 107–114. https://doi.org/10.52958/iftk.v19i2.6149

Riswandi Ishak, Setiaji, Fajar Akbar, & Mahmud Safudin. (2020). Rancang bangun sistem informasi surat masuk dan surat keluar berbasis web menggunakan metode waterfall. *Jurnal Indonesia Sosial Teknologi*, *1*(3), 198–209. https://doi.org/10.36418/jist.v1i3.33

Saputra, O., Fitri, I., & Handayani, E. T. E. (2022). Sistem pakar diagnosa kerusakan hardware komputer menggunakan metode forward chaining dan certainty factor berbasis website. *Jurnal JTIK (Jurnal Teknologi Informasi dan Komunikasi)*, *6*(2), 234–242. https://doi.org/10.35870/jtik.v6i2.416

Sastypratiwi, H., & Nyoto, R. D. (2020). Analisis data artikel sistem pakar menggunakan metode systematic review. *JEPIN (Jurnal Edukasi dan Penelitian Informatika)*, *6*(2), 250-257. https://doi.org/10.26418/jp.v6i2.40914.

Surya Pratama, H., Putri, M., Roby, M., & Tusakdiyah, S. H. (2022). Sistem pakar diagnosa kerusakan laptop atau komputer menggunakan metode forward chaining. *JEKIN - Jurnal Teknik Informatika*, *2*(1), 16–23. https://doi.org/10.58794/jekin.v2i1.100

Taufiq, R., & Sandi, A. P. (2021). Perancangan sistem pakar diagnosa kerusakan laptop dengan penerapan metode forward chaining. *JIKA*, 260–264.

Wahid, A. A. (2020). Analisis metode waterfall untuk pengembangan sistem informasi. *J. Ilmu-ilmu Inform. dan Manaj. STMIK, no. November*, *1*(1), 1-5.

Waang Bler Tuang, S., Elefri Neno, F., & Dappa Ege, E. (2024). Penerapan metode Naïve Bayes untuk diagnosa kerusakan komputer. *JATI (Jurnal Mahasiswa Teknik Informatika)*, *7*(4), 2636–2640. https://doi.org/10.36040/jati.v7i4.7710

Widianto, S. C., Widada, B., Sandradewi, K., & Remawati, D. (2025). Implementasi metode certainty factor dalam sistem. *Jurnal*, *13*(1), 40–49.