Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

June 2026. Vol. 5. No. 3; e-ISSN: 2808-4519

Techdoctor Android Application Design for Laptop Damage Diagnosis Based on Symptoms Experienced by Users

Ahmad Zulfan Hafiz Harahap[1]*, Zulfahmi Indra[2], Tegas Ramadhan[3], Hafizh Ariq[4]

[1,2,3]*Universitas Negeri Medan*

Ahmadzulfann2021@gmail.com[1*], zulfahmi.indra@unimed.ac.id[2], kuy410zml@gmail.com[3], zalfaameutia@gmail.com[4]

Abstract

Laptop damage is a common problem faced by computer users in today's digital era. Users generally have difficulty identifying the source of damage due to limited technical knowledge. This research aims to design and build an Android application named TechDoctor that helps users perform initial diagnosis of laptop damage independently based on selected symptoms. The application was developed using the Kotlin programming language in the Android Studio environment and applies a rule-based expert system approach with forward chaining inference method. The knowledge base was compiled from literature studies and interviews with experienced laptop technicians, covering common symptoms such as overheating, battery failure, LCD problems, and RAM or hard disk issues. Testing was carried out using the Black Box Testing method with 30 test scenarios, yielding a diagnosis accuracy rate of 86.7%. The results show that the TechDoctor application is capable of providing initial diagnosis and solution recommendations quickly, accurately, and in a manner easily understood by non-technical users.

Keywords: *Android; Diagnosis; Expert System; Kotlin; Laptop damage; Rule-based system*

1. Introduction

Laptops are one of the most widely used technological devices supporting educational and professional activities. However, users often face technical problems they cannot identify independently due to limited technical expertise [1]. In many cases, users resort to bringing their devices to professional repair services for issues that could potentially be resolved without expert intervention, resulting in unnecessary time and financial costs.

A rule-based expert system is a computational approach that encodes expert knowledge in the form of IF-THEN rules to support decision-making. According to Kusrini [2], an expert system is a system that adopts human knowledge into a computer so the computer can solve problems as experts do. This approach is highly relevant for developing a laptop damage diagnosis tool that can guide non-technical users through a structured diagnostic process.

Several previous studies have explored this domain. Chrystianto and Sumardi [3] developed an Android-based expert system for hardware troubleshooting using backward chaining, while Dahri et al. [4] implemented a web-based forward chaining system for laptop diagnosis, reporting satisfactory accuracy. The present work differs in that TechDoctor targets a mobile Android platform with a user-friendly interface accessible to non-technical end users.
This research aims to design and build the TechDoctor Android application as a digital tool that enables users to perform preliminary laptop damage diagnosis independently and efficiently.

2. Literature Review

2.1. Expert systems

An expert system is a computer program designed to simulate the capabilities and knowledge of a domain expert. Hartati and Iswanti [5] define an expert system as a system that attempts to transfer human expertise into a computer so that it can solve problems usually only resolved by specialists. The core components of an expert system include: (1) knowledge base, (2) inference engine, (3) user interface, and (4) explanation facility.

Forward chaining is an inference strategy that begins from known facts and applies rules to derive conclusions. It is particularly suitable for diagnostic applications where users provide observed symptoms and the system determines the most probable cause [6]. This method was selected for TechDoctor because of its natural alignment with the symptom-selection workflow.

2.2. Android platform and Kotlin language

Android is a Linux-based mobile operating system developed by Google, representing the world’s most widely used mobile OS [7]. Kotlin is a modern, statically typed programming language developed by JetBrains and officially endorsed by Google as the preferred language for Android development since 2019 [8]. Compared to Java, Kotlin offers more concise syntax, improved null safety, and full support for functional programming paradigms [9].

2.3. Common laptop failures

Laptop failures can be categorized into hardware and software damage. The most commonly encountered hardware failures include: (1) overheating caused by a dirty cooling fan or dried thermal paste; (2) battery damage indicated by rapid discharge or inability to charge; (3) LCD damage such as a striped, cracked, or blank screen; and (4) RAM or hard disk failure causing frequent system hangs or Blue Screen of Death (BSOD). Early identification of these symptoms is critical to prevent further deterioration.

3. Research Methodology

This research employs the waterfall software development method, comprising five sequential phases: (1) requirements analysis, (2) system design, (3) implementation, (4) testing, and (5) maintenance. The waterfall model was selected due to the well-defined and stable requirements identified at the outset of the project, consistent with the guidelines outlined by Pressman [10].

3.1. Requirements analysis

Requirements analysis was conducted through a literature review and structured interviews with three experienced laptop technicians. This process yielded 25 failure symptoms mapped to 8 primary damage categories. The knowledge acquisition approach adopted follows the elicitation methodology described by Pressman [10] for rule-based knowledge systems.

3.2. Knowledge base design

The knowledge base was formulated using IF-THEN rules, where each rule maps one or more symptom combinations to a specific diagnosis. For instance: IF [laptop overheats] AND [fan noise is loud] THEN [Diagnosis: Overheating — clean cooling fan and replace thermal paste]. A total of 32 rules were derived from the expert interviews and validated against documented repair cases.

3.3. System architecture

TechDoctor was designed using the Model-View-ViewModel (MVVM) architecture pattern, which is Google’s recommended approach for modern Android development. MVVM separates business logic from the user interface layer, resulting in a codebase that is more structured, testable, and maintainable. The inference engine was implemented in Kotlin using an in-memory Map data structure to store and match rules in real time.

4. System design and implementation

4.1. Application structure

The TechDoctor application consists of five main screens: (1) Splash Screen, displaying the application logo and name on startup; (2) Home, providing the main navigation menu; (3) Diagnosis, the core screen presenting the selectable symptom list; (4) Result, displaying the diagnosis outcome and recommended corrective actions; and (5) About, containing developer information and a usage guide.

4.2. Inference mechanism

The forward chaining inference engine processes symptom selections in real time. Each time a user adds or removes a symptom, the engine re-evaluates all rules against the current fact set. If a rule’s condition set is fully satisfied, the corresponding diagnosis is activated and displayed on the Result screen along with a prioritized list of recommended repair steps.

Table 1: Font specifications for A4 papers

Font Size (pt)	Application	Style
8	Table/figure captions, reference items	Regular, centered
9	Author affiliation, abstract body	Regular/Bold (headings L3)
10	Author names	Bold, centered; L2 heading left
12	Section headings (L1)	Bold, left-justified
18	Paper title	Bold, centered

5. Results and discussion

5.1. Black box testing

System testing was performed using the Black Box Testing method. Thirty test scenarios were prepared, each representing a distinct symptom combination mapped to an expected diagnosis. The application’s output was compared against the expected diagnosis provided by the expert technicians.

Table 2: Black box testing results of TechDoctor application

Damage Type	Test Cases	Passed	Accuracy
Overheating	6	6	100%
Battery failure	5	5	100%
LCD damage	5	4	80%
RAM problem	6	5	83.3%
Hard disk problem	5	4	80%
Software problem	3	2	66.7%
Total	**30**	**26**	**86.7%**

5.2. Discussion

As shown in Table 2, TechDoctor achieved perfect accuracy (100%) for overheating and battery failure scenarios, as the symptoms associated with these categories are highly specific and non-overlapping. Lower accuracy was observed for software problems (66.7%) due to the ambiguous nature of software symptoms, which can manifest similarly across multiple damage types.

Overall, the 86.7% accuracy rate is considered satisfactory for a rule-based expert system. A comparable study by Taufik and Sandi [11] on a laptop damage diagnosis expert system using forward chaining reported an accuracy of 85%, confirming that TechDoctor’s performance exceeds the benchmark of similar research.

6. Conclusion

This research successfully designed and built the TechDoctor Android application as an expert-system-based tool for preliminary laptop damage diagnosis. The application was developed using Kotlin on Android Studio, implementing a rule-based forward chaining inference engine with a knowledge base covering 25 symptoms across 8 damage categories. Black Box Testing with 30 scenarios yielded an overall accuracy of 86.7%, demonstrating the application’s reliability for non-technical users.

Future work should extend the knowledge base to cover additional damage types including software-related failures, apply probabilistic methods such as Certainty Factor to improve accuracy in ambiguous cases, and integrate an online update mechanism to keep the knowledge base current with evolving laptop technologies.

Acknowledgment

The authors would like to thank the experienced laptop technicians who contributed their knowledge during the expert interviews, as well as colleagues who assisted in testing the application. This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

[1] S. Nurajizah and M. Saputra, "Sistem pakar berbasis Android untuk diagnosa kerusakan komputer," Jurnal Pilar Nusa Mandiri, vol. 14, no. 2, pp. 129-136, 2018.
[2] Kusrini, Sistem Pakar, Teori dan Aplikasi. Yogyakarta: Penerbit Andi, 2006.
[3] H. Chrystianto and I. Sumardi, "Sistem pakar troubleshooting kerusakan hardware laptop dengan metode backward chaining berbasis Android," Jurnal Ilmiah Infrastruktur Teknologi Informasi (JIITI), vol. 2, no. 1, pp. 9-15, 2021.
[4] S. Dahri, H. Basri, I. K. Annafiyah, T. Yulistio, A. Saifudin, and I. Kusyadi, "Sistem pakar mendiagnosa kerusakan laptop untuk membantu menemukan masalah berbasis web menggunakan metode forward chaining," Jurnal Teknologi Sistem Informasi dan Aplikasi, vol. 4, no. 4, pp. 268-273, 2021, doi: 10.32493/jtsi.v4i4.14772.
[5] S. Hartati and S. Iswanti, Sistem Pakar dan Pengembangannya. Yogyakarta: Graha Ilmu, 2008.
[6] R. Taufik and A. P. Sandi, "Perancangan sistem pakar diagnosa kerusakan laptop dengan penerapan metode forward chaining," JIKA (Jurnal Informatika), vol. 5, no. 2, 2021, doi: 10.31000/jika.v5i2.4598.
[7] N. Safaat, Pemrograman Aplikasi Mobile Smartphone dan Tablet PC Berbasis Android, Edisi Revisi. Bandung: Informatika, 2018.
[8] M. S. Talib, "Kotlin programming language: A comprehensive overview of definition, applications, advantages, and limitations," Int. J. Comput. Appl. Technol. Res., 2024, doi: 10.7753/ijcatr1303.1002.
[9] D. Nurlaila and T. Susanti, "Comparative analysis of Kotlin and Java programming language in memory usage," JISCO (Journal of Information System and Computing), vol. 1, no. 1, pp. 1-6, 2022.
[10] R. S. Pressman, Software Engineering: A Practitioner's Approach, 8th ed. New York: McGraw-Hill Education, 2015.
[11] R. Taufik and A. P. Sandi, loc. cit.
[12] I. Sommerville, Software Engineering, 10th ed. Boston: Addison-Wesley, 2016.