**Kielce University of Technology**

**Faculty of Electrical Engineering, Automatic Control   
and Computer Science**

**Mustafa Efe Emeksiz**

**Student Identification Number: 95330**

**Information Service About Computer Hardware Faults**

**A thesis for   
a first-cycle qualification**

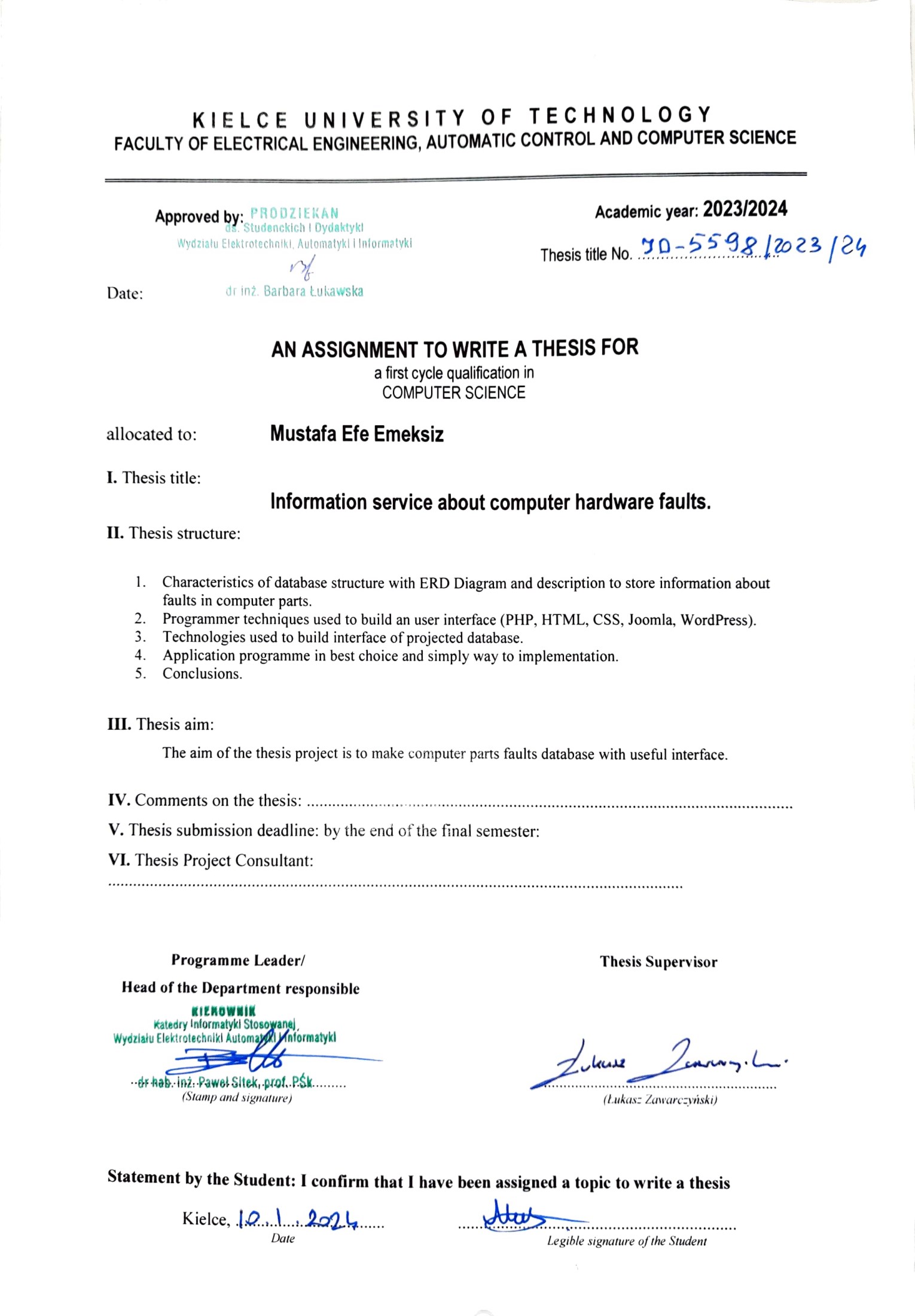
**Programme of study: Computer Science**

Thesis Supervisor:

dr inż. Łukasz Adam Zawarczyński

Department of Computer Science Applications

Kielce, 2024



metin, mektup, harf, doküman, belge, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Information Service About Computer Hardware Faults**

**Abstract**

This thesis details the creation of an information service designed to address computer hardware faults. It introduces an integrated system for reporting and predicting hardware issues, constructed with PHP scripts and robust database management practices. The system's strength lies in its analytical capabilities, allowing for the examination and graphical representation of fault data, which can reveal potential error patterns and highlight hardware components at greater risk of failure. Additionally, the user-friendly interface, built on the WordPress platform, enhances accessibility for end-users, streamlining the process of fault management. This study not only sheds light on common computer malfunctions but also advances ideas for preemptive maintenance strategies.

Keywords: - Databases, Information Systems, WordPress, PHP, Xampp, Statistics

TABLE OF CONTENTS

[1. Introduction 11](#_Toc158686569)

[1.1 Motivation 11](#_Toc158686570)

[1.2 Importance of Fault Reporting 11](#_Toc158686571)

[1.3 Significance of Statistical Analysis 12](#_Toc158686572)

[1.4 Informed Decision-Making 12](#_Toc158686573)

[2 Similar applications 13](#_Toc158686574)

[2.1 IBM PMQ 13](#_Toc158686575)

[2.2 Dell ProSupport Plus for PCs 14](#_Toc158686576)

[2.3 Hitachi Predictive Maintenance 14](#_Toc158686577)

[3 Database 15](#_Toc158686578)

[3.1 ERD 15](#_Toc158686579)

[3.2 Clarification of the Data 17](#_Toc158686584)

[3.3 Utilization of PhpMyAdmin in Database Management System 20](#_Toc158686594)

[4. Server 22](#_Toc158686595)

[4.1 Introduction to XAMPP as a Local Server Solution 22](#_Toc158686596)

[5 Using WordPress For Information Service About Computer Hardware Faults 24](#_Toc158686597)

[5.1 What is Wordpress 24](#_Toc158686598)

[5.2 Logic Behind Choosing WordPress 24](#_Toc158686599)

[5.3 WordPress Themes and Plugins 25](#_Toc158686600)

[6. Building a Custom Dataset for Fault Reporting Analysis 29](#_Toc158686605)

[6.1 Using Python For Building a Custom Dataset 29](#_Toc158686606)

[7. Data Analytics and Visualization in Computer Hardware Faults Information System 35](#_Toc158686613)

[7.1 User Interface: Fault Report Insert Form 35](#_Toc158686614)

[8. Conclusion 52](#_Toc158686623)

[9. References 53](#_Toc158686624)

**Abbreviations List**

**CPU**  Central Processing Unit

**PC**  Personal Computer

**ERD**  Entitiy Relationship Diagram

**PMQ**  Predictive Maintenance and Quality

**RAM**  Random Access Memory

**IT**  Information Technology

**SQL**  Structured Query Language

**HDD**  Hard Disk Drive

**CMS**  Content Management System

**GPL**  General Public License

**HTML**  HyperText Markup Language

**SSD**  Solid-State Drive

**NVMe**  Non-Volatile Memory Express

**SATA**  Serial Advanced Technology Attachment

**PHP**  Hypertext Preprocessor

**GIF** Graphic Interchange Format

**PNG**  Portable Network Graphics

**JPEG**  Joint Photographic Experts Group

**WBMP** Wireless Application Protocol Bitmap Format

**XPM**  X PixMap

# 1. Introduction

## 1.1 Motivation

In the contemporary landscape of technological advancements, where the seamless operation of computer systems is integral to organizational success, the effective management of faults is a critical imperative. As organizations grapple with an increasing dependence on computer systems, the timely identification, reporting, and resolution of faults becomes paramount. The motivation behind this research lies in addressing the challenges inherent in fault management and leveraging statistical analysis to extract actionable insights. I believe numbers and statistics are always trying to point out such facts. By working on such a project, I plan to find and represent statistics and patterns that might help with better management. This structure can also represent low-quality or low-durability computer parts that cost extra to the company back in the days up to the dataset we have.

## 1.2 Importance of Fault Reporting

In the realm of computer systems, timely identification and reporting of faults are pivotal to prevent operational disruptions and mitigate risks associated with data loss and system downtime. This thesis emphasizes the use of historical fault data to trace patterns that can influence cost-effectiveness and decision-making regarding hardware investments. After some review of statistics and dataset, it can give information on the reliability of different CPU brands, offer insights into cost versus performance, and identify the potential for efficiency enhancements.

Through a meticulous examination of fault occurrences, this research provides a foundation for predictive analytics, facilitating more informed choices in hardware maintenance and procurement.

## 1.3 Significance of Statistical Analysis

Statistics plays a pivotal role in this study, serving as a powerful tool to decipher patterns, trends, and correlations within the collected data. Through statistical analysis, we aim to uncover valuable insights into the dynamics of fault reporting systems. Statistical techniques provide a quantitative lens through which we can identify the prevalence of specific fault types, discern reporting patterns over time, and assess the efficacy of resolution strategies.

## 1.4 Informed Decision-Making

The motivation to incorporate statistical analysis lies in its capacity to empower organizations with data-driven decision-making. By elucidating statistical trends within fault reporting, this research aims to provide decision-makers with a robust foundation for implementing targeted strategies to enhance fault management. Statistical insights offer a  proactive approach to system reliability, allowing organizations to preemptively address potential issues and optimize their fault resolution mechanisms.

In conclusion, the motivation for this research rests on the imperative to contribute actionable insights to the field of fault reporting. By leveraging statistical analysis, we strive to offer a comprehensive understanding of the fault reporting process, thereby empowering organizations to fortify their systems and sustain optimal performance in an ever-evolving technological landscape.

# 2 Similar applications

In today's world, I think it's nearly impossible to find an application that no one ever thinks of. But of course, it is possible to come up with a better design, a more specific approach, or more functionalities for the existing ones. When I started this project, I couldn’t find any applications or ideas of similar applications; however, when I finished it and did more research, I found some applications made by companies.

## 2.1 IBM PMQ

With IBM® Predictive Maintenance and Quality, you can monitor, analyze, and report on information that is gathered from devices. In addition, recommendations for actions can be generated by Predictive Maintenance and Quality [6]. I searched online for some user interface screenshots, but I couldn’t find too many. I guess this is the general problem with these applications. They are not very popular and are meant for very small groups of people and companies. However, Figure 2.1 below represents a screenshot of the application's interface for a customer. This interface might give us some ideas about how it looks for a better comparison between my interface.

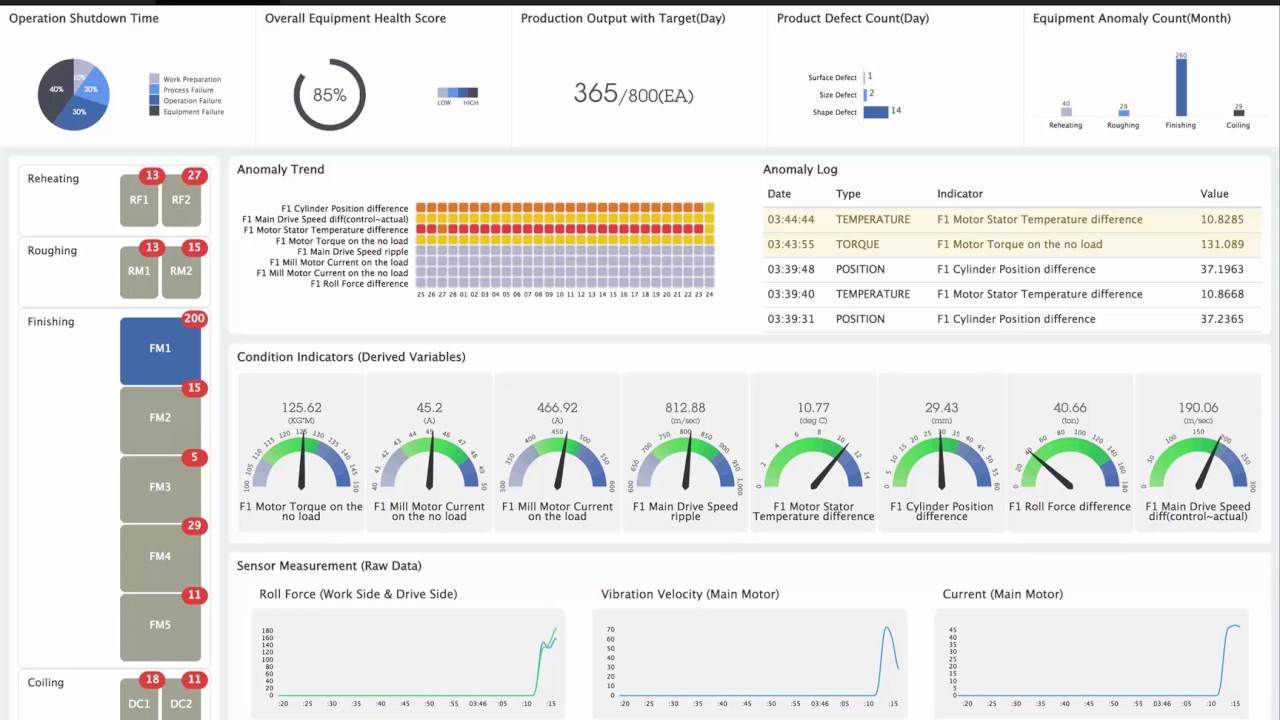


Figure 2.1 - Representing the Thumbnail of Video posted by The Mainstream Seer [9]

## 2.2 Dell ProSupport Plus for PCs

Another application that I found is Dell ProSupport Plus for PCs. Just like IBM’s application, there is information related to it on their pages, and as I understand it, it's a  specific application that they provide to their customers through payment. It's most likely for customer support, but it has a feature called “Predictive Failure Analysis Enabled by  SupportAssist.”. Which is related to my application.

## 2.3 Hitachi Predictive Maintenance

Hitachi Predictive Maintenance helps practitioners evaluate assets and diagnose problems using monitoring tools like sensors and image analytics. We leverage advanced algorithms and machine learning principles to deploy repeatable maintenance solutions and

an optimized asset lifecycle maintenance plan [7]. Just like the other applications that I  presented, this one is also for specific customers, and I can’t even find images related to  their application’s interface because they do the application according to the customer's requirements. Figure 2.2 below explains their point of view on these features and presents a reasonable cycle that can also give us some ideas about our structure and the possible benefits of our application.

metin, ekran görüntüsü, yazı tipi, logo içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 2.2 - Representing the Image Taken From www.hitachi.us [10]

# 3 Database

The chasm between having a basic idea of what your database needs to be able to do and designing the appropriate tables is bridged by having a clear data model. Data modeling involves thinking very carefully about the different sets or classes of data needed for a particular problem [5].

## 3.1 ERD

metin, ekran görüntüsü, diyagram, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 3.1 - Representing the ERD

In the Computer Faults Information System, the relationships between entities capture the complex interactions of components and faults within computer hardware. These relationships are essential for understanding how fault reports are tied to specific hardware components and how these components are part of a larger computer system.

### FaultReport to Computer Tables Relationship

The faultreport entity has a one-to-many relationship with the computer entity, established through the ComputerID attribute. This relationship signifies that multiple fault reports can be associated with a single computer, reflecting the real-world scenario where a computer may experience various faults over time.

### Component Relationships

Each computer is further linked to its individual components through a series of one- to-one relationships, indicating that each computer record includes references to specific parts such as RAM, CPU, Motherboard, PowerSupply, GraphicsCard, HardDrive, and ExternalCPU. These relationships are defined by the corresponding ID fields within the computer entity, such as MotherboardID and PowerSupplyID. These one-to-one relationships suggest that for every individual computer, there is a distinct record for each of its internal components within the database.

### Implications for Fault Analysis

The structured relationships between the computer and its components enable detailed fault analysis. By examining the linked fault reports, technicians can identify patterns in faults, such as recurring issues with specific components, and can also track the fault history of a particular computer.

### Why We Choose This ERD

The chosen ERD structure is designed to mirror the functional requirements of the Computer Faults Information System, allowing for comprehensive tracking of hardware issues. The central faultreport entity captures instances of hardware malfunctions, which are then meticulously linked to the computer entity, reflecting the real-world scenario where computers can exhibit multiple, distinct faults over their lifespan. This central relationship is supported by additional one-to-one linkages between the computer entity and various hardware component entities such as RAM, CPU, motherboard, etc., which catalog the specific parts that could be implicated in each fault report. These relationships are pivotal in diagnosing recurring issues, streamlining maintenance processes, and facilitating detailed hardware component tracking within the organizational IT infrastructure. The ERD’s entity relational structure is designed for database normalization, promoting data integrity, and reducing redundancy, which is essential for maintaining an accurate and efficient fault reporting system.

## 3.2 Clarification of the Data

In the design of our fault reporting system, careful consideration was given to the selection of data tables to ensure a comprehensive and efficient representation of  fault- related information. The following key data tables were chosen based on their relevance to the fault reporting process and the need for a structured and organized database schema. I have a total of nine tables, and I will show each table in the following subheadings. I have done research about computer parts and attributes of computer parts for more realistic results and better accuracy. Based on my research, I created the tables below.

### Faultreport Table

metin, yazı tipi, sayı, numara, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 3.2 - Representing the Structure of Faultreport Table

The central table in our database schema is the Faultreport table, which captures crucial information related to each reported fault. This includes fields such as ReportDate for the date of the report, Status indicating the current status of the fault, ComputerID to uniquely identify the affected computer system, FaultType specifying the nature of the fault, and Description providing detailed information about the reported issue. This table forms the foundation for fault data management and facilitates efficient tracking and resolution. As the last field, we have ReprortID for tracking the fault easier, and for each entry of faults, we have it auto-increment itself. Figures from Figure 3.3 to Figure 3.10 present attributes of  my database tables and give us a general view of our database structure. By understanding that, we can assume what functionalities we can use, and it might also give some ideas to people who want to create similar structures.

### Computer Table

metin, yazı tipi, sayı, numara, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 3.3 - Representing the Structure of Computer Table

The computer table serves as a cornerstone in our database, capturing overarching information about individual computer systems. Fields such as ComputerID, Brand, and Model allow for unique identification and classification. Including this table enhances fault tracking by associating reported issues with specific hardware configurations. For the first 59 computers, we inserted our server data. We trace MotherboardID, PowersupplyID, GraphicscardID, and ExternalCPUID for only servers.

### CPU Table

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 3.4 - Representing the Structure of CPU Table

Unlike the previous tables, these other tables will represent the components that we will scope in my work. They only contain features of the components, which is why I will just mention their structures with brief explanations.

### ExternalCPU Table

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 3.5 - Representing the Structure of ExternalCPU Table

### Graphicscard Table

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 3.6 - Representing the Structure of Graphicscard Table

### Harddrive Table

metin, yazı tipi, ekran görüntüsü, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 3.7 - Representing the Structure of Harddrive Table

### Motherboard Table

metin, yazı tipi, çizgi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 3.8 - Representing the Structure of Motherboard Table

### Powersupply Table

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 3.9 - Representing the Structure of Powersupply Table

### Ram Table

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 3.10 - Representing the Structure of Ram Table

## 3.3 Utilization of PhpMyAdmin in Database Management System

PHPMyAdmin is a series of scripts that you can use in conjunction with Apache and PHP to administer your Webserver [4]. In the development of the Computer Faults Information System, phpMyAdmin served as the primary interface for managing the MySQL database. As a free and open-source administration tool, phpMyAdmin offers a web-based graphical user interface that simplifies the management of database activities. This tool facilitated the creation, modification, and deletion of databases, tables, and indexes, thereby streamlining the process of database design.

metin, ekran görüntüsü, yazılım, bilgisayar simgesi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 3.11 - Representing The PhpMyAdmin Interface

Through phpMyAdmin, complex SQL queries were executed with ease, allowing for an overview of stored data and the fine-tuning of the database structure without the need for command-line interactions. It has a user-friendly interface that allows you to perform most operations either through queries or buttons specified for those operations. As you can see in the figure, which is represented above, I created my database called ComputerFaults, which includes all of my tables related to WordPress and the database itself. We can also display the ERD and table structure through a graphical interface.

# 4. Server

## 4.1 Introduction to XAMPP as a Local Server Solution

XAMPP is a completely free, easy-to-install Apache distribution containing MariaDB, PHP, and Perl [1]. I found it worth mentioning that it is much safer to install the above-mentioned applications separately for a better configuration. In my case, I don't have any problems with it, but it's much safer, for sure. It is open source, just like all of the other futures that I used during my project, and that makes my project more available to anyone who wants to run my project. It is pretty easy to configure, and XAMPP contains both MySQL and Apache, which I need to use for my project. Figure 4.1 will represent the XAMPP control panel that we will use for our application, and I will explain some details using this interface.

metin, ekran görüntüsü, yazılım, bilgisayar simgesi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 4.1 - Representing the XAMPP Control Panel

As you can see, it is pretty easy to reach out to the configuration files. Also, we can start both our database and Apache using the same interface. The green highlight means that they are both running fine without any problems. During my configuration, my computer was already using Port 80, which is the default port for Apache. That’s why I changed Apache ports to 443 and 4433. in Figure 4.2, we will check the configuration file.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 4.2 - Representing the http.conf File for the Configuration

As we can see in the last line, I have changed 80 by 4433. Normally, it’s not necessary to change the ports for initializing Apache. You can either keep it as default or you can choose any port that you want. Because of that change, for displaying my pages on my browser, I need to add a small addition at the end of the localhost domain. Which will look like the screenshot that I will put below. In Figure 4.3 we will see how to access Mainpage.php using our browser.



Figure 4.3 - Representing the http.conf File for the Configuration

It is a PHP page that I tried with my PHP snippets. I used WordPress for my project because of the reasons and features that I will explain later in my thesis. Choosing XAMPP also gives me another advantage, which is XAMPP’s compatibility with WordPress.

# 5 Using WordPress For Information Service About Computer Hardware Faults

## 5.1 What is WordPress

WordPress serves as a CMS, providing a user-friendly platform that enables website organization and design with minimal need for coding and technical development. Today, WordPress is built on PHP and MySQL, and licensed under the GPLv2. It is also the platform of choice for over 43% of all sites across the web [2].

## 5.2 Logic Behind Choosing WordPress

WordPress is a user-friendly application that is available to everyone, just as I mentioned before. Additionally, it has countless themes and plugins. Which will give me the freedom to route my project the way that I imagined. Since it’s open source and a very common platform, there are too many resources available on the internet. It helped me find the plugins and the solutions to the bugs a lot easier due to its huge community. My project mostly contains PHP and HTML snippets. That’s why it's also possible to do the same operations without using WordPress. But there will be much more to be done for the styling of my pages and other possible details about design. In Figure 5.1, I will represent the WordPress interface for more familiarity with the interface and better understanding.

metin, ekran görüntüsü, yazılım, web sayfası içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.1 - Representing the WordPress Welcome Page

## 5.3 WordPress Themes and Plugins

### WordPress Themes

WordPress offers a huge amount of themes for users who want to create their webpages. Since my project is an information service, I chose the theme “Twenty Nineteen” which used to be WordPress's default theme for 2019. It looks simple and easy to edit for my specific requirements.

### Plugins

This is the feature of WordPress that makes it possible for me to use WordPress. There are a huge number of plugins for too many different operations. I used a plugin called XYZ PHP Code, which you can also find by writing Insert PHP Code Snippet. This plugin was developed by a developer named xyzscripts.com. I need it for running my PHP snippets on my WordPress page, and I will give more details when I mention PHP snippets.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.2 - Representing the Icon of Plugin

It is also quite easy to use, and we can name our PHP snippets by headers; after that, we insert the PHP snippets inside. By doing that, we create short names for the PHP snippets, and we can call these short names on the page where we want to run them.

metin, sayı, numara, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.3 - Representing the Interface of the Plugin

We can also edit and delete the short names and the PHP snippets inside them.



Figure 5.4 -Representing How to Call the Shortnames on the Webpage

Another plugin that I used was done by the same developer. This time, it's for inserting HTML snippets. The plugin called Insert HTML Snippet can also be found by writing XYZ HTML.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.5 - Representing the Icon of Plugin

Since it has been done by the same developer that I talked about, it has the same interface as the plugin that I mentioned before. Very easy to use. I just called the short names, and HTML snippets are executed in the part of the page that I call it.

The other two plugins that I used for a better experience while using WordPress are one of them for dark mode, and the other one is classic editor. Since they are not essential, I’m not going to explain more about them.

### How to Download and Integrate WordPress

Firstly, we need to download the latest version of XAMPP. Then we put this zip file into the C:\xampp\htdocs folder destination, and we unzip the file. Just like we will see on Figure 5.6.

metin, ekran görüntüsü, yazılım, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.6 - Representing the Unzipped WordPress File

An important thing worth mentioning is that you need to create the database first to be able to use WordPress. After all of these steps, we need to go to the localhost/wordpress destination. Since I've already done this configuration, I can’t show it, but you can see the steps that will be asked below. After doing these steps, we can use WordPress. In Figure 5.7, I will represent all the steps that are necessary and that are provided by WordPress for a general look at what steps we will go through.

metin, ekran görüntüsü, iş kartı, logo içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.7 - Representing the WordPress Steps

### Security Features of WordPress

During the configuration, we set a username and password for our WordPress. Each time we log out if we try to google the localhost/wordpress domain it will direct us to the login page. You can see Figure 5.8 below, which represents that.

metin, ekran görüntüsü, yazı tipi, logo içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5.8 - Representing the WordPress Login Page

# 6. Building a Custom Dataset for Fault Reporting Analysis

We can consider this thesis in two parts. The first part was generally about design, configurations, and how to do everything before coding. Starting with this chapter, we can finally focus on what I have done. After clarifying the topic and doing all the configurations, I realized I couldn’t find any datasets that suited my work. I searched online data creator tools and open-source data repositories, but I couldn’t find anything. That’s why I decided to create my own Python algorithms for the creation and insertion of my data.

## 6.1 Using Python For Building a Custom Dataset

Remember that a program is just a sequence of instructions telling a computer what to do. Obviously, we need to provide those instructions in a language that a computer can understand [8]. Python's versatility and user-friendly nature make it an ideal language for furthering my programming skills. Having a solid foundation in Python, I conducted additional research to develop specialized algorithms for each table in the database, facilitating the creation of structured data. So let’s start with the faultreport table, which is one of the main tables in my dataset. I will just mention important code snippets from the whole structure.

### Crafting Data Entries for the Faultreport Table

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.1 - Representing the Date Generation for Fautreport Table

The random\_date function in the Python code snippet is designed to generate a random datetime object. By calculating the total seconds between a given start\_date and end\_date, the function uses the random.randint method to select a random number of seconds within this interval. It then creates a timedelta object with this random number of seconds and adds it to the start\_date, resulting in a random datetime that falls between the two specified dates. This function is crucial for simulating realistic date entries in the dataset for the fault reports table. This snippet is used in all tables that need a date attribute.

metin, yazı tipi, ekran görüntüsü, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.2 - Representing the Date Range for Faut Report Table

For the generation of fault entries, we chose dates between 2010 and 2023 for this test, but we can also manually add other years. I will show this manual data insert functionality later.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.3 - Representing the Possible Error Entries

The generate\_description function in the snippet dynamically assigns a fault description based on the type of hardware fault. It does this by selecting a random description from a predefined list associated with each fault type, such as 'RAM' or 'CPU'. This allows for the generation of varied and realistic fault descriptions for each record in my dataset, which is crucial for simulating a wide range of potential issues in the fault reporting system. I have done research for possible errors in each computer part and created a list of them. Then I inserted them in this algorithm, as you can see in the figure above. I would rather have real data that I took from some company or a free data repository, but since that’s my only opportunity, this research makes my dataset realistic and reliable.

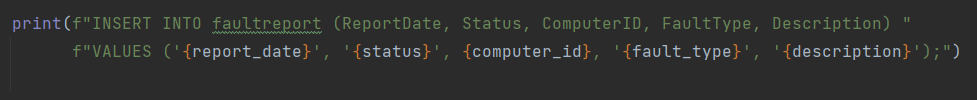


Figure 6.4 - Representing the Print Statement

After the creation of the data, inserting it into the SQL database will not take a very long time since it's not formatted by our Python script output. This way, we can just copy and paste the output, and the data will be inserted.

ekran görüntüsü, metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.5 - Representing the output of Faultreport Table

The output you see in Figure 6.5 only happened on Earth, and maybe it's never going to happen in that order again due to the huge pool of data lists, dates, and order of the data. That allows us to create an unlimited number of entries for my project. For my test, I have entries around 1300. This print statement and output structure are used in all other tables.

### Crafting Data Entries for the Computer Table



Figure 6.6 - Representing the code Snippet of Computer Table

This single line of code that you can see in the figure generates a random integer between 100000 and 999999, ensuring a 6-digit number. The f preceding the string indicates

a formatted string literal, known as an f-string, which allows the inclusion of the result of expressions within string literals. The random.randint function is crucial here as it ensures that each serial number is likely to be unique by selecting from a pool of 900,000 possible numbers. The 'SN' prefix is a static string that gets concatenated with the random number to form the serial number for a computer record, following a common convention for serial number formats.



Figure 6.7 - Representing the Code Snippet of Computer Table

The initial 50 entries in the dataset were designated as servers to reflect their greater system capacities. This distinction was made to simulate an inventory with realistic specifications, where servers typically have more robust capabilities compared to standard user computers.

metin, yazı tipi, ekran görüntüsü, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.8 - Representing the Code Snippet of Computer Table

Most computer’s purchase dates are between 2007 and 2010. There was less data for such computers after 2010 I generally keep the dates between these years to make sure there are no errors before the computer is purchased. But of course, if I want to extend it later, it's quite easy to add newer computers. Also, by keeping the entries for computers for 3 years, we will have a huge scale of dates for the faultreport entries, which is the most important part of my work. If any of the computers are replaced by new ones, we can overwrite them on the database, but this is not a functionality that I focused on since I created the working dataset for my thesis and there’s no such case needed.

### Crafting Data Entries for the Motherboard Table

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.9 - Representing the Code Snippet of Motherboard Table

In the dataset creation process for the motherboard table, a distinctive approach was employed to ensure that each motherboard brand is associated with a realistically corresponding model. For instance, while ASUS has the 'Prime B450' model, MSI is correctly paired with the 'Tomahawk B550' model, adhering to real-world product lineups. This mapping prevents the generation of data with improbable brand-model combinations, preserving the authenticity of the simulated inventory. The randomized selection still occurs within the bounds of these predefined pairings, introducing variability while maintaining a logical structure. This was a problem that I faced during my planning, and I solved it that way.

### Crafting Data Entries for the HDD Table

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.10 - Representing the Code Snippet of Motherboard Table

The code snippet represents a critical part of the data generation process for hard drives, focusing on the accurate assignment of interfaces. It implements a conditional logic structure to differentiate between the interfaces used by SSDs and HDDs. If the drive is an SSD, the script randomly chooses between the 'NVMe' and 'SATA' interfaces to reflect the variety of connection options available for SSDs on the market. Conversely, if the drive is an HDD, the script assigns the 'SATA' interface by default, as it is the standard interface for HDDs. This thoughtful approach ensures that the dataset accurately represents the technological differences between the two types of storage devices.

### Crafting Data Entries for the Powersupply Table

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.11 - Representing the Code Snippet of Powersupply Table

This part of the code ensures that each power supply unit in my dataset looks realistic. For each brand, I pick a model and a voltage that match what's available. This way, my data reflects the true variety of power supplies out there.

By structuring our data with these dictionaries, we ensure coherent attribute pairing across our dataset, despite the random selection process.

### Explanation of the CPU,GPU and RAM Table: A Glimpse into Data Structuring and Algorithmic Consistency

Starting from this table, I have similar data structures and algorithms. That’s why I want to show the data pools of my tables briefly. In Figures 6.12, 6.13, and 6.14, we will take a look at this.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.12 - Representing the Code Snippet of CPU Table

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.13 - Representing the Code Snippet of GPU Table

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6.14 - Representing the Code Snippet of RAM Table

By showing all the code snippets, I tried to show that I had done detailed research and created algorithms for each table that covered the requirements for the functionalities that would be needed in the future.

# 7. Data Analytics and Visualization in Computer Hardware Faults Information System

## 7.1 User Interface: Fault Report Insert Form

This PHP snippet that I will show was designed for a WordPress template page that allows users to insert fault reports into a database. The code is responsible for handling user input from a form, validating and sanitizing the input, and then securely inserting the data into the database. Upon successful insertion, it provides immediate feedback to the user. By using such functionality, we will be able to insert data without using phpmyadmin. I will show the working PHP snippet that does data insertation in Figure 7.1.

metin, ekran görüntüsü, yazı tipi, doküman, belge içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.1 - Representing the Code Snippet of Data Insertation Page

The PHP snippet is responsible for processing form data submitted via the POST method. It starts by checking if the form has been submitted ($\_SERVER["REQUEST\_METHOD"] == "POST"). It then proceeds to validate and sanitize the input to ensure that all the required fields are filled out by the user. This is crucial to prevent the submission of incomplete forms, which can lead to database errors or inconsistencies.

The isset() function checks if a variable is set and is not NULL, and this check is performed for each expected POST variable. If any required field is missing, it sets the $message variable to an error string that prompts the user to fill in all required fields, enhancing the user experience by providing immediate feedback on the submission.

Finally, if all required fields are present, the script employs a prepared statement to insert the data into the database. Prepared statements are a robust feature provided by PHP to prevent SQL injection, a common security vulnerability. By using placeholders (?) in the SQL query and binding the actual input values with $stmt->bind\_param(), the code safely inserts user data into the database. I will show what the insertion page looks like in Figure 7.2 for a good representation of what the whole code does.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.2 - Representing the Test Entry

Upon executing the prepared statement with $stmt->execute(), the script checks for success. If the execution is successful, it sets a success message; otherwise, it catches the error and sets an appropriate error message. The $stmt->close(); statement is essential for releasing the prepared statement and freeing up resources, which is a good practice for efficient memory usage and preventing potential leaks. I have to add all these functions later because I was trying the functionalities on another PHP page before I put them on my WordPress page. That’s why when I tried to put it on my WordPress page, data wasn’t inserted, and some of the methods that I mentioned here helped me a lot with debugging. I tested its functionality manually, and everything worked well, just like I represented in Figures 7.3 and 7.4.

metin, yazı tipi, grafik, logo içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.3 - Representing the massage that I showed in code snippet



Figure 7.4 - Representing the Result of Our Test on PhpMyAdmin

### SQL Query Optimization for Fault Pattern Recognition

The PHP script I will present in Figure 7.5 is designed to connect to a MySQL database and execute a complex SQL query that identifies the computer with the most frequently reported fault type, along with related information such as the total number of errors, the date of the latest error, and the shortest time difference between consecutive errors for that fault type.

metin, ekran görüntüsü, yazı tipi, doküman, belge içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.5 - Representing the Code Snippet

The innermost subquery uses the LAG function to get the previous ReportDate for each fault report, partitioned by ComputerID. This means that for each computer, report dates are compared to find the time difference between faults.

The middle subquery groups these results by ComputerID and FaultType, counting the total number of errors (TotalErrors), finding the most recent error date (LatestErrorDate), and the shortest time difference between consecutive errors (ShortestTimeDifference), which is calculated using the DATEDIFF function between ReportDate and LagReportDate.

Finally, the outer query selects the computer that has the highest number of errors and the shortest time between errors. It orders the results by TotalErrors in descending order and ShortestTimeDifference in ascending order, and limits the result to just one record. This returns the computer that is most prone to errors, with its most common fault type and related information. Figure 7.6 will represent the output, which represents the result of the SQL query execution

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.6 - Representing the Output of The Code

The computer with ID 59 was identified as having the most errors, with RAM issues being the most common fault. It has reported a total of 11 errors, with the latest being on 2023-05-23. The 'ShortestTimeDifference' being 0 indicates that there have been multiple instances of fault reports on the same day, suggesting frequent occurrences  of issues with the RAM. This insight could be pivotal for further investigation into the quality of components or usage patterns that could be contributing to these frequent failures. Understanding these patterns is crucial for improving system reliability and maintenance strategies.

### Dynamic Data Visualization for Computer Inventory

This PHP script in Figure 7.7 facilitates dynamic data visualization on a web page. It connects to a MySQL database, fetches statistical data, processes it, and finally displays it using Chart.js. The core of the script lies in transforming database query results into a JSON format that Chart.js can interpret to render bar charts. With this setup, the chart displayed on the webpage reflects current data from the database, allowing for an up-to-date and interactive data analysis experience.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.7 - Representing the Code Snippet

In this code snippet, PHP prepares the data for visualization by encoding it into JSON, a format compatible with JavaScript. This data is then utilized within a JavaScript block, where Chart.js is invoked to render a bar chart on the web page. The chart is configured to display on the HTML canvas identified by 'computerChart'. This seamless integration of server-side data handling with client-side chart rendering exemplifies modern web application techniques for dynamic data presentation. In Figure 7.8, I will represent how this code displays a chart on my WordPress page.

metin, ekran görüntüsü, diyagram, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.8 - Representing the Chart

Originally, my project was not intended to include JavaScript snippets; however, during my research, I discovered that using JavaScript not only simplified performance but also significantly enhanced the visual appeal of the application. Using it for real-time updated charts is quite efficient.

### Analyzing Fault Report Distributions and Error Intervals

This PHP snippet identifies the fault types, counts their occurrences, pinpoints the computer with the most errors, and calculates the period between the first and last reported errors. If the query is successful, the data is presented in an HTML table. If the database connection fails, an error message is displayed, and the script is halted. After processing, the database connection is closed to conserve resources. It is pretty common checking that I have used in other snippets too. In Figure 7.9, I will represent a small part of the code and explain it after that.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.9 - Representing the Code Snippet

This snippet represents the core logic for aggregating fault reports. It identifies the fault type (FaultType), counts the number of occurrences of each fault type (FaultTypeCount), identifies the computer with the most errors of each fault type (ComputerWithMostErrors), and calculates the shortest interval between errors for each fault type (DaysBetweenErrors). This SQL query is designed to provide a comprehensive overview of the fault data, which can be used for further analysis and decision-making in the context of fault reporting systems. The Figure 7.10 will display a breakdown of faults recorded in our database.

metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.10 - Representing the Output on the WordPress Page

It shows the frequency of each fault type. It also highlights the computer that experienced the most occurrences of a particular fault. Additionally, it reveals the shortest interval between repeated instances of the same fault, providing a snapshot of error recurrence rates. This knowledge can tell us about possible patterns or mistakes made in our dataset.

### Distribution of Fault Types in Computer Systems

I want to represent the common fault types using a pie chart. After some research, I found that it is possible to use PHP, but I need to use a library called GD. PHP is not limited to creating just HTML output. It can also be used to create and manipulate image files in a variety of different image formats, including GIF, PNG, JPEG, WBMP, and XPM. Even more conveniently, PHP can output image streams directly to a browser. You will need to compile PHP with the GD library of image functions for this to work [3].

Normally I have already explained the general configuration, but since I need to enable it during my experiments, I found it worth mentioning how to enable it briefly.

### Congifuration of GD Library

I opened the php.ini file using the xampp interface that I will present in Figure 7.11. You will also see other figures between Figures 7.12, 13, 14, and 15 for a better understanding and visualization of the configuration supporting the explanations that I made.

metin, ekran görüntüsü, yazı tipi, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.11 - Representing the XAMPP Interface

We need to find the line extension=gd and uncomment it.

metin, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.12 - Representing the php.ini

To check if its working or not, we need to create a PHP page called php.info.

metin, ekran görüntüsü, sayı, numara, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.13 - Representing the htdocs Folder

Inside of the PHP file need need to put single line phpinfo().

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.14 - Representing the Code Snippet

After that, we go to the phpinfo domain, and we need to see something like that to see if the GD library is working. metin, ekran görüntüsü, çizgi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.15 - Representing the php.info Page

I found this configuration worth mentioning because there were fewer sources about its configuration on Windows. Also, even if I configured it correctly, there was no way to check if it was working properly or if my code was correct. That’s why I explained how to configure and check if it is working or not briefly. I didn’t explain it in the configuration part since it's not for general use in my work.

### Source Code Snippet of the Pie Chart

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.16 - Representing the PHP Snippet

This code snippet is responsible for generating a pie chart using PHP's GD image functions. Initially, the script sets up the dimensions for the image and allocates colors for the chart and text. It then fills the background with white. For each fault type from the fetched data, it calculates the slice angle based on the proportion of the count to the total and draws the pie chart segment using the imagefilledarc() function. The colors for each segment are randomly generated, providing a unique color for each fault type. Following the chart, a legend is created to help identify each fault type with its corresponding color in the chart.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.17 - Representing the How to Call the Snippet

The HTML is for embedding a PHP-generated image into a webpage. The <img> tag displays an image that is generated by a PHP script (piechart.php). The src attribute of the <img> tag is set to the URL where the PHP script is hosted. When the webpage loads, the browser sends a request to this URL, and the PHP script executes and generates the pie chart image, which is then sent back to the browser to be displayed on the page.

This approach allows me to dynamically generate images based on real-time data from a database. Instead of a static image file, the source is a PHP script that can change the output depending on the data it processes. This is useful for displaying up-to-date information in a graphical format, like a pie chart, that updates automatically when the underlying data changes. It's an efficient way to incorporate interactive data visualization into the webpage without requiring a page refresh or additional user input. I will show this pie chart in Figure 7.18, which is displayed on my WordPress page using the snippet that I created.

metin, renklilik, ekran görüntüsü, daire içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.18 - Representing the Output

As you can see, it shows the percentages of each error that happened to the computer components until today. It automatically renews itself each time that I open the WordPress page. I could also try to do it by using Javascript to make this pie chart look better, but this fulfills my requirements and expectations, so that’s why I don’t use it.

### Fault Report Retrieval by Computer ID

Another functionality that I added was checking the fault history of a computer by giving its ID on the user interface. This functionality is quite important, in my opinion, because besides the statistics part of my work, it points out the historical data tracing part of my work too.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.19 - Representing the Code Snippet

This code snippet above uses PDO to securely fetch fault reports for a specific computer from the database. It prepares a parameterized SQL query to prevent SQL injection, executes it with a user-provided computer ID, and then retrieves the results as an associative array for easy access and display. The process provides us with strong and secure database interaction in a PHP application.

metin, yazı tipi, ekran görüntüsü, logo içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.20 - Representing the Test

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.21 - Representing the Test Result

This functionality is also tested manually due to the large number of computers in  my database. This computer with computerID 101 used to have zero entries when I tried it in the first place. After I added a faultreport using my interface, we can display the history of reports assigned to computerID 101.

### Predictive Analysis of Fault Occurrences

The last functionality that I added to my work is a prediction for the next possible error. It’s quite possible if we think that there are such possibilities for founding patterns. In Figure 7.22, I will show my code snippet for predicting the next possible error by their occurrence frequency. Of course, there are other ways of predicting it. For example, machine learning will be a much better option for this operation. However, for machine learning, my dataset might be considered small for accurate predictions, and my dataset is complicated for such predictions due to the huge number of parameters. I tried an application called Orange Data Mining, but it didn't work the way I wanted. Also, there are some ways to use AI tools like ChatGPT for those operations by just giving them the dataset, and AI will do the rest of the operations, but I don't find it a good option.

metin, yazı tipi, ekran görüntüsü, cebir içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.22 - Representing the Code Snippet

We are grouping the result by faulttype first. After grouping, it orders the results in descending order (ORDER BY Frequency DESC) to ensure that the fault type with the highest count is at the top of the list. Finally, the LIMIT 1` clause ensures that only the most frequent fault type is retrieved. Figure 7.23 and Figure 7.24 I will show you what this code displays on my WordPress page.



Figure 7.23 - Representing the Output on WordPress

metin, ekran görüntüsü, yazılım, işletim sistemi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.24 - Representing the Output on WordPress

Just as we can see, it will do the whole operation in the background and find the highest frequency of error. Then it will put it between the sentences that I prepared. There are also too many possible ways to represent this output. Since it's just the computer part that has the highest chance of having the next possible error, I chose a single-line sentence for better understanding by the end users. We can see that for my experiment, the Graphicscard has the highest frequency of errors, which tells us that probably the next error will be caused by it.

### General look to the WordPress page

As I showed earlier, you have seen all of the functionalities that my work provides. Now we can look at how it looks on my WordPress page. I separated the data insert page and the statistics page for a better experience, just like I will show in Figures 7.25 and 7.26.

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.25 - Representing the WordPress Page

metin, ekran görüntüsü, yazılım, web sayfası içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.26 - Representing the WordPress Page

Thanks to WordPress, styling my work was easy. If we want to do everything without using WordPress, we also need to use CSS way more than we used it in my case. I will show a general look at my webpage in Figures 7.27, 7.28, and 7.29. For a better understanding of how WordPress helped me organize my webpage, in Figure 7.30, I will show a screenshot of the creation of my top menu for my pages. I didn't need to use a single line of code for this operation. As we will see in all the screenshots below, the styling and readability are in the way that I wanted. Of course, in the future, if I want to add more functionalities, I can also change the order and the approach that I use right now.

metin, ekran görüntüsü, yazılım, web sayfası içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.27 - Representing the WordPress Page

metin, yazılım, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.28 - Representing the WordPress Page

metin, yazılım, bilgisayar simgesi, web sayfası içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.29 - Representing the WordPress Page

yazılım, multimedya yazılımı, metin, bilgisayar simgesi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7.30 – Representing How to Create Menu for the WordPress Page

As we can see, I have developed an information system about computer hardware faults. This gives us historical data, making predictions and providing us statistics that, if someone who knows has taken a look at them, might help us develop preemptive technologies. I will talk about these in the conclusion part. Of course, I didn't put all of my pages here, but we had a general look at them.

# 8. Conclusion

This thesis encapsulates the journey of creating a robust information service platform for diagnosing and predicting computer hardware faults. The system's backbone, constructed through sophisticated PHP scripting and database management techniques, has demonstrated a proficient capacity for logging, analyzing, and visualizing hardware fault data. This facilitation provides a granular view of system vulnerabilities and an informed basis for preventive strategies. A notable achievement of this project is the successful integration of dynamic SQL query generation and PHP-driven server-side processing to anticipate potential fault occurrences.

Throughout the project, a comprehensive understanding of the fault reporting mechanism was gained, and the platform was aligned with user-friendly principles, employing WordPress as its framework. While the current state of the application stands as a testament to the practical application of PHP in fault diagnosis and predictive analytics, there is recognition of the potential for further enhancement. Incorporating more detailed statistical data analysis could refine the predictive accuracy of the system. Additionally, a collaboration with a real-world organization to utilize their proprietary data could offer a valuable opportunity to scale and test the application in a live environment. Such an endeavor would not only validate the system's efficacy but also provide invaluable insights into its real-world adaptability and scalability.

The experience has laid the foundation for ongoing learning and set the stage for future exploration. The next steps could include developing machine learning algorithms that can learn from historical data to provide even more accurate predictions or integrating real-time data feeds for dynamic fault assessment. This work could prove invaluable for businesses looking to proactively manage their hardware systems, optimize performance, and mitigate risks associated with hardware failures.

# 9. References

[1] XAMPP. (n.d.). Retrieved December 15, 2023, from <https://www.apachefriends.org/>

[2] WordPress. (n.d.). Retrieved December 15, 2023, from <https://wordpress.org/about/>

[3] PHP: Image Processing and Generation. (n.d.). Retrieved January 14, 2024, from <https://www.php.net/manual/en/intro.image.php>

[4] Norman, M. (2004). Database Design Manual: using MySQL for Windows. Springer. p.206.

[5] Churcher, C. (2012). Beginning Database Design From Novice to Professional. Apress. p.23.

[6] Predictive Maintenance and Quality. (2021-03-05). <https://www.ibm.com/docs/en/pm-and-q/2.6.1?topic=deployment-predictive-maintenance-quality>

[7] HITACHI. (n.d.). Retrieved January 17, from <https://www.hitachivantara.com/en-us/pdf/hcc/solution-profile/hitachi-predictive-maintenance.pdf>

[8] Zelle, J. (2016). Python Programming: An Introduction to Computer. FRANKLIN BEEDLE & ASSOC. p.6

[9] The Mainstream Seer. (27 March 2017). Retrieved January 25, 2024. IBM PMQ Manufacturing Demo. <https://www.youtube.com/watch?v=HRix556u0KE>

[10] Image from the Hitachi America, Ltd. Research & Development webpage, accessed on 25 January. <https://www.hitachi.us/rd/solutions/predictive-maintenance.html>