A SYNOPSIS ON

Software Resource Manager for Embedded Devices

Submitted in partial fulfillment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

In

Computer Science & Engineering

Submitted by:

Neha Kanwal-2261389

Lokesh Rathour-2261336

Lokesh Rawat -2265010

Bhawana Mehta-2261141

Under the Guidance of
Mr. Prince Kumar
Assistant Professor

Project Team ID: 47

Department of Computer Science & Engineering

Graphic Era Hill University, Bhimtal, Uttarakhand

March-2025

Graphic Era Hill University
BHIMTAL CAMPUS

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the Synopsis entitled **"Software Resource Manager for Embedded Devices "**in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science & Engineering of the Graphic Era Hill University, Bhimtal campus and shall be carried out by the undersigned under the supervision of **Mr.Prince Kumar, Assistant Professor**, Department of Computer Science & Engineering, Graphic Era Hill University, Bhimtal.

Neha Kanwal	2261389
Lokesh Rathour	2261336
Lokesh Rawat	2265010
Bhawana Mehta	2261141

The above mentioned students shall be working under the supervision of the Mentor signed on the Project Topic **"Software Resource Manager for Embedded Devices"**

Supervisor
(Mr. Prince Kumar)

Head of the Department
(Dr. Ankur Bisht)

Internal Evaluation (By DPRC Committee)

Status of the Synopsis: Accepted☐ Rejected ☐

Any Comments:

Name of the Committee Members: **Signature with Date**
1.
2.

Table of Content

Chapter 1

Introduction And Problem Statement

1.1 Introduction

In today's digital era, computing devices are becoming increasingly integral to our daily lives. From personal computers to embedded systems, we depend on them for work, communication, entertainment, and countless other tasks. However, not all users have access to high-end devices with abundant computational power and memory. A large section of the population, particularly in developing regions or rural areas, still relies on low-end laptops, smartphones, or embedded systems that struggle with performance issues due to limited hardware capabilities. These constraints often result in slow operation, frequent lags, system crashes, and overall poor user experience.

Low-end devices are generally characterized by low processing power (e.g., older Intel or ARM CPUs), limited RAM (1–2GB), and restricted storage space (e.g., 16–32GB eMMC). While they may be sufficient for basic operations, even moderate multitasking can strain the device, leading to sluggish performance. Applications or background processes consume a significant amount of system resources, often running unnecessarily and clogging up memory or CPU cycles. Additionally, improper file management leads to cluttered storage, further degrading the system's responsiveness.

To address these challenges, we propose the development of a Smart Resource Manager (SRM) — a lightweight software solution specifically designed to optimize the performance of low-end computing devices. The SRM tool aims to manage CPU, RAM, and storage resources in real time and improve system performance without needing hardware upgrades.

What is Smart Resource Management?

Smart Resource Management refers to the intelligent monitoring and optimization of system resources such as the CPU, memory, disk, and processes. By constantly tracking resource usage, the system can automatically adjust, terminate non-essential processes, clear memory, and organize storage to ensure smooth performance. It uses lightweight algorithms and techniques to minimize its own resource consumption while improving the overall efficiency of the system.

The SRM application will consist of two major components:

1. A Lightweight OS Optimizer – Adjusts CPU scheduling, RAM allocation, and storage cleanup in real-time.
2. An Automated File Management System – Organizes, compresses, and secures files automatically based on usage frequency and type.

By combining these features, SRM enhances the usability of outdated or underpowered devices, allowing them to remain functional for longer durations.

Why is SRM Needed?

SRM (Smart Resource Manager) is essential for low-end devices because it optimizes limited resources like CPU, memory, and storage to ensure smooth performance and prevent system slowdowns or crashes. By intelligently managing background tasks, prioritizing critical processes, and reducing unnecessary resource usage, SRM improves responsiveness, enhances battery life, and extends the overall lifespan of the device. This makes it possible for resource-constrained systems to run efficiently and provide a better user experience despite hardware limitations.

With the rapid advancement in software requirements and system updates, older or budget devices often cannot keep up. Applications demand more memory and storage; operating systems grow heavier; and background services multiply with every installed app. For users who cannot afford to upgrade hardware, SRM provides a cost-effective solution by squeezing more performance out of existing devices.

1.2 Problem Statement

With the increasing dependence on digital technology, the performance and efficiency of computing devices have become more important than ever. While high-end systems are capable of handling intensive applications and multitasking with ease, low-end devices—such as budget laptops, entry-level smartphones, and embedded systems—often struggle to keep up with modern software demands. These devices, due to their limited CPU power, RAM capacity, and storage space, are prone to performance degradation, especially when running multiple applications or background processes.

Many users across the world, especially in developing countries, rely on such low-end devices for daily computing needs, including education, remote work, communication, and entertainment. These users frequently face issues such as:

- System slowdowns during multitasking.
- Lag or freezing when switching between applications.
- Crashing applications due to memory overflow.
- Low storage space, which hinders updates and new installations.
- Poor system responsiveness even during basic operations like browsing or file handling.

One of the major reasons behind these issues is inefficient resource management. Modern operating systems and applications often assume the presence of adequate system resources and therefore fail to perform efficiently on older or low-spec hardware. Background processes, startup applications, and system services consume significant resources without the user's knowledge or control. Moreover, users often lack the technical expertise or time to manually monitor system performance, clear unnecessary files, or optimize running processes.

Existing resource management tools available in the market are typically designed for more powerful systems and may themselves consume excessive resources. Furthermore, they often lack automation, require manual intervention, and are not tailored to the specific constraints of low-end systems. This leads to a gap where a lightweight, intelligent, and user-friendly tool is needed to optimize system performance without adding to the system load.

- Therefore, there is a clear and pressing need for a solution that:
- Monitors system resource usage (CPU, RAM, storage) in real-time.
- Automatically detects and optimizes high-resource processes.
- Cleans up junk files and manages storage efficiently.
- Provides users with control over unnecessary background applications.
- Operates with minimal system overhead and works smoothly on low-end devices.

The proposed project, Smart Resource Manager (SRM), aims to address this problem by creating a software tool that provides real-time resource monitoring, process management, and automated file optimization, all within a lightweight and easy-to-use interface. The tool is specifically built to enhance performance on low-end devices, thereby extending their usability and improving user experience.

This project will not only benefit individual users but also support institutions and communities that rely on older hardware, promoting digital inclusivity, cost-efficiency, and sustainable technology use.

Challenges Faced by Low-End Devices:

Thermal Throttling: Low-end CPUs often overheat and reduce clock speeds to cool down, slowing performance even further.

Limited Multithreading Support: Most budget processors have fewer cores and threads, resulting in slower multitasking.

Non-SSD Storage: Many low-end systems still use HDDs, which are slower in read/write speeds compared to SSDs, affecting system boot time and application launch speed.

Battery Optimization Conflicts: Aggressive battery-saving features in low-end mobile devices often kill background processes, affecting system consistency.

Gaps in Existing Solutions:

Lack of real-time optimization: Most tools only clean temporary files or suggest recommendations instead of actively optimizing performance.

User-unfriendly interfaces: Many tools are designed for advanced users with technical knowledge, leaving average users confused.

Resource-heavy software: Ironically, some system optimizers themselves consume high CPU/RAM usage, defeating the purpose.

No predictive management: Current tools don't analyze usage patterns to predict and optimize future behavior.

Why SRM is Needed:

Custom-built for lightweight operation: Designed to use minimal resources itself while maximizing device efficiency.
Real-time process monitoring and control: Helps users easily kill unnecessary or misbehaving tasks.
Automated file system cleanup: Keeps storage free by identifying junk, unused files, and duplicates.

Modular architecture: Allows flexibility for features like scheduled optimization, secure file management, and performance tracking over time.

Impact of Smart Resource Manager:

Better device lifespan: Reduces performance stress on aging hardware.

Enhanced user experience: Smooth performance for basic tasks like web browsing, video playback, or document editing.

Supports education and small businesses: Especially useful for organizations unable to afford regular hardware upgrades.

Chapter 2

Background/Literature Survey

2.1 Background

In recent years, the global reliance on digital systems has grown significantly. However, not all users have access to modern, high-spec devices. A considerable portion of the population—especially in developing nations, rural areas, or educational institutions—still relies on low-end or older computing devices. These systems often suffer from hardware limitations, including slower processors, minimal RAM, limited storage, and outdated operating systems. As software requirements continue to evolve, these devices struggle to meet the performance expectations of modern users.

2.1.1. Rising Dependence on Digital Devices

With the rapid evolution of technology, access to digital devices has become essential for education, work, healthcare, and communication. While high-end systems dominate the market, a significant portion of the population still depends on low-end or outdated computers, especially in developing countries, rural schools, government offices, and low-income households. These devices are generally equipped with:
Low RAM (1GB–4GB)
Older or slower processors (dual-core or early quad-core) Limited
storage (HDDs or small SSDs)
Legacy operating systems (Windows 7, XP, or lightweight Linux distros)
These systems struggle to run modern software, leading to frequent crashes, system lags, and application unresponsiveness.

2.1.2. Performance Issues in Low-End Device Low-end devices often face:

High CPU usage due to background processes and bloated apps,Memory leaks from software that doesn't release RAM properly.Fragmented storage, which slows read/write operations,Startup bloat, where unnecessary apps launch during boot, increasing boot time.

2.1.3. Environmental and Economic Impact

Replacing slow devices with newer models is not always feasible:
Environmentally, it leads to e-waste accumulation, which contributes to pollution and resource depletion.Economically, many users or institutions cannot afford frequent upgrades.
Therefore, prolonging the life of older systems through intelligent optimization tools is a sustainable and cost-effective solution.

2.1.4. Importance of Resource Management

Resource management is a core aspect of operating systems, where CPU scheduling, memory allocation, and storage control ensure efficient performance. However, default resource managers are not proactive and often lack:Automated decision-making,Intelligent process prioritization,Predictive cleanup mechanisms,Low-end systems need smarter, real-time solutions that can make these decisions on behalf of the user with minimal manual intervention.

Low-end devices often become laggy, unresponsive, or unstable when used for basic daily tasks such as browsing the web, watching videos, or running multiple applications simultaneously. With limited technical knowledge, most users are unaware of how to manage system resources, disable background services, or clean up unnecessary files manually. These limitations highlight the need for an automated solution to manage system performance intelligently and efficiently without user intervention.

The Smart Resource Manager (SRM) project addresses these issues by proposing a lightweight, modular, and intelligent system that dynamically monitors and optimizes CPU, RAM, and storage usage. The project is focused on enhancing performance and extending the usable life of low-end systems through automation and simplicity.

Literature Survey

2.2.1 Existing Operating System Resource Managers

Modern operating systems such as Windows, Linux, and Android offer built-in tools for monitoring system resources. Examples include Windows Task Manager, Linux's top or htop commands, and Android's internal memory manager. These tools display system performance metrics but are not always optimized for low-end systems.
They often lack:Real-time optimization,Automated cleanup or process termination,Predictive analysis based on usage behavior,Intuitive user interfaces for non-technical users,While they are helpful, they are passive monitors rather than active managers, leaving optimization tasks to the user.

2.2.2 Third-Party Optimization Tools

There are several third-party system optimizers available on the market:
CCleaner,Advanced SystemCare,Clean Master (Android),Glary Utilities
These tools offer features like:Junk file cleaning,RAM optimization,Startup management,Registry cleaningHowever, these applications often have drawbacks:High system resource usage themselves
Advertisement-heavy interfaces in free versions,Manual operation required for most tasks,No process-level optimization or task killing,Limited compatibility with embedded or Linux-based devices
They also do not adapt to system usage patterns or provide long-term system behavior predictions. Most are general-purpose optimizers, not tailored for low-end systems, making them inefficient for resource-constrained environments.

2.2.3 Research Papers and Studies

Several academic and industry studies have analyzed the performance bottlenecks in low-end devices and proposed solutions.
Kang, J., et al. (2018) explored adaptive memory management for Android-based smartphones, emphasizing garbage collection and memory compaction for low-memory devices.
Abdul Rahman, H., et al. (2019) investigated lightweight process scheduling techniques for resource-limited systems and found that dynamic prioritization based on real-time metrics could enhance performance.
Singh, P., et al. (2020) developed a framework for storage optimization in mobile devices that automatically cleaned cache and duplicate files.
Kim, H., & Lee, D. (2021) introduced predictive CPU throttling using usage pattern analysis to prevent performance degradation in low-end systems.
These studies confirm the importance of adaptive, intelligent, and low-overhead resource management techniques in resource-constrained environments.

2.2.4 Embedded Systems and IoT Optimization

The techniques used in embedded systems and Internet of Things (IoT) devices have inspired parts of this project. These devices often have extremely limited resources, and developers use event-driven programming, real-time scheduling, and memory-efficient file systems to ensure efficiency.
Techniques like:
Process prioritization,Event-based scheduling,Memory footprint reduction,Lightweight data compression.

2.2.5 Gaps in Current Literature and Solutions

Despite several advancements, the following gaps still exist:Lack of unified solution combining real-time monitoring, process control, and file optimization.
Absence of lightweight, cross-platform tools suitable for both desktop and embedded systems. No
focus on user simplicity and accessibility in system optimization tools.
Limited research on predictive optimization, which can prevent problems before they arise.

Chapter 3

Objectives

3. Objective

The primary objective of this project, Smart Resource Manager for Low-End Devices, is to develop a lightweight, efficient, and user-friendly software tool that can enhance the performance of outdated or resource-constrained systems. The following objectives outline the specific goals of the project in detail:

3.1. Optimize System Resources in Real-Time

The system aims to monitor and manage critical resources such as CPU usage, RAM utilization, and disk space in real-time. Low-end devices often suffer from limited processing capabilities and low memory, which can quickly become overwhelmed by modern applications and background processes. The Smart Resource Manager will intelligently assess the resource usage and take automated actions like clearing memory cache, terminating unresponsive background processes, and adjusting resource allocation dynamically to maintain system stability and responsiveness. This objective ensures better performance even when the hardware is not powerful.

3.2. Automate File Management and Storage Optimization

Another core objective is to design a robust file management system that automatically organizes, compresses, and secures user data. Often, users are unaware of storage clutter caused by duplicate files, temporary files, unused applications, and large media files. The Smart Resource Manager will provide features such as junk file cleanup, scheduled folder organization, file compression to save space, and encryption for sensitive data. This will help users manage storage effectively without manual intervention and extend the usability of systems with limited disk capacity.

3.3. Improve Usability through Intuitive User Interfaces

One of the key challenges with most system optimization tools is their complexity, which can be intimidating for non-technical users. To address this, the Smart Resource Manager will include both Graphical User Interface (GUI) and Command Line Interface (CLI) versions tailored for simplicity and accessibility. The UI will feature a dashboard displaying real-time system metrics, one-click optimization buttons, and process management features. This objective ensures that even users with minimal technical knowledge can benefit from the tool.

3.4. Extend the Lifespan of Older Devices

By improving the operational efficiency of low-end systems, this project aims to prolong their useful life and delay the need for hardware replacement. Many individuals, schools, and organizations operate on limited budgets and cannot afford frequent upgrades. A well-optimized system can run smoothly even on older hardware, thus reducing e-waste, saving costs, and promoting sustainable use of technology. This aligns the project with environmental responsibility and economic efficiency.

3.5. Ensure Lightweight Design and Compatibility

The final objective is to ensure that the tool itself is lightweight and compatible with older versions of operating systems like Windows 7/8, lightweight Linux distributions, and systems with minimal system specifications. The software will be developed using resource-efficient libraries and will avoid heavy background tasks. This makes it feasible to run on legacy systems without causing additional system load, which is crucial for achieving meaningful performance improvements.

Chapter 4

Hardware and Software Requirements

4.1 Hardware Requirements:

Sno.	Name of hardware	Specification
1	Processor	Intel Core i3or higher
2	RAM	Minimum 4 GB
3	Hard Disk	Minimum 250 GB
4	Display	Standard 15.6'' Monitor
5	Input Devices	Keyboard and Mouse
6	Network Adapter	Ethernet/Wi-Fi enabled

4.2 Software Requirements:

Sno.	Name of Software	Specification
1	Operating System	Windows 7/8/9/10
2	Programming Language	Python 3.2/C
3	IDE or Code Editor	VS Code/Pycharm
4	Libraries & Frameworks	tkinter ,matplotlib
5	Documentation Tool	MS Word/Latex

Chapter 5

Possible Approach / Algorithms

In the domain of intelligent systems and data-driven applications, the selection of appropriate algorithms is critical for optimizing performance, accuracy, and overall system reliability. This chapter presents an in-depth discussion of the possible algorithmic approaches applicable to the proposed system. The focus is primarily on clustering and feature selection techniques, utilizing fuzzy logic and optimization strategies to improve results.

5.1 Feature Selection Techniques

Feature selection is a crucial preprocessing step in machine learning and pattern recognition tasks. The goal is to select a subset of relevant features from the original set to reduce dimensionality, enhance model performance, and avoid overfitting. One common metric used for evaluating feature selection is the Root Mean Squared Error (RMSE).

RMSE Equation:

$$RMSE = \sqrt{\{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2\}/\{n\}}$$

Where:
p_1 = actual value
q_1 = predicted value
n = number of data points

Lower RMSE values indicate better feature selection and prediction accuracy.

5.2 ABC Algorithm (Artificial Bee Colony)

The ABC algorithm is a swarm-based metaheuristic optimization algorithm inspired by the foraging behavior of honey bees. It has shown promising results in clustering, optimization, and data classification tasks.

Input:
D : Dataset
K : Number of clusters

Output:
Optimal cluster centers Z
Membership matrix U

5.3 Pseudo Code for ABC Algorithm

Begin
1. Initialize Z by choosing k points from D randomly;
2. Initialize W with wij = 1 ($1 \leq j \leq k$, $1 \leq h \leq d$);
3. Estimate U from initial values of W and Z using Equation 2.7;
4. Let error = 1 and Obj = Eαcl(W,Z);

5. while error > 0 do
a. Update Z according to Eq. 2.6;
b. Update W according to Eq. 2.5;
c. Update U according to Eq. 2.7;
d. Calculate NewObj = Eαcl(W,Z);

Let error = |NewObj – Obj|, and then Obj <= NewObj
6. End while

This iterative algorithm ensures the continuous improvement of clustering by minimizing the objective function , which is a function of weights and cluster centers.

Filter Method for Feature Selection

Filter methods evaluate the relevance of features by their intrinsic properties without involving any machine learning algorithms. These methods are computationally efficient and are suitable for high-dimensional datasets. A popular approach in filter-based selection is using mutual information, chi-square, or correlation coefficients to rank features.

5.4 Evaluation and Optimization

The algorithm is validated using various evaluation metrics such as:

RMSE (Root Mean Squared Error)
Accuracy
Silhouette Score (for clustering)
Time Complexity
The optimization ensures that the final model not only performs well but is also scalable and efficient.

References

1. Ghosh, R., & Naik, V. (2012). *Bottleneck detection and performance improvement for resource management in cloud environments*. In *2012 IEEE Fifth International Conference on Cloud Computing* (pp. 456-463).

2. Chiang, R. H., & Huang, H. (2011). *TRACON: Interference-aware scheduling for data-intensive applications in virtualized environments.* In *Proceedings of the 2011 ACM Symposium on Cloud Computing* (pp. 1-14).

3. Verma, A., Pedrosa, L., Korupolu, M., Oppenheimer, D., Tune, E., & Wilkes, J. (2015). *Large-scale cluster management at Google with Borg*. In *Proceedings of the Tenth European Conference on Computer Systems* (pp. 1–17).

4. Mao, M., Li, J., & Humphrey, M. (2010). *Cloud auto-scaling with deadline and budget constraints*. In *2010 11th IEEE/ACM International Conference on Grid Computing* (pp. 41–48).