A SYNOPSIS ON

CUSTOMIZED TASK MANAGER

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

BACHELOR OF TECHNOLOGY

In

Computer Science & Engineering

Submitted by:

Atul Joshi	**2261121**
Aakriti Garkoti	**2261050**
Priyanshu Kumar	**2261444**
Shivam Tiwari	**2261529**

Under the Guidance of

Mr. Anubhav Bewerwal

Assistant Professor

Project Team ID: 110

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 **"Customized Task Manager"** in partial fulfilment 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 **Guide Name, Designation**, Department of Computer Science & Engineering, Graphic Era Hill University, Bhimtal.

Atul Joshi	2261121
Aakriti Garkoti	2261050
Priyanshu Kumar	2261444
Shivam Tiwari	2261529

The above mentioned students shall be working under the supervision of the undersigned on the **"Customized Task Manager"**

Signature
Supervisor

Signature
Head of the Department

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 Contents

Chapter 1

Introduction and Problem Statement

1. Introduction

In the modern era of computing, effective system resource management is critical to the smooth functioning of computers. All computer systems execute multiple applications and background processes concurrently, thereby necessitating the monitoring and control of system performance in real time. Task manager is an indispensable system utility that allows users to examine CPU utilization, memory usage, running processes, and overall system condition.

Windows includes an inbuilt Task Manager that helps users to analyze system performance and close stuck processes. Nonetheless, it lacks numerous limitations, including a dearth of customization options, limited process management functions, and a very limited set of data visualization functions. Further, most users view the stock Windows Task Manager GUI as being too simple or far too complex depending on the individual's level of technical skill.

To overcome these challenges, a tailored task manager has been created with:

1. .NET Framework for application development to maintain compatibility with Windows platforms.
2. C# (C-Sharp) as the programming language to support effective management of system-level tasks.
3. XAML (Extensible Application Markup Language) for developing an easy-to-use and visually pleasing user interface.
4. WPF (Windows Presentation Foundation) to design a rich-featured desktop application with dynamic UI components.
5. Visual Studio as the main development environment for coding, debugging, and testing.

The suggested task manager presents system performance metrics in real time, such as CPU usage, memory use, and full process control. It presents an interactive and easy-to-use interface that simplifies system resource monitoring and controlling for both technical and non-technical users. The application features more visualization, improved filtering capabilities, and new functionalities like sorting processes by CPU/memory use and real-time graphically viewed system performance.

A task manager is not only a performance monitor, but it is also an essential system utility for resolving problems like CPU spikes, memory leaks, and frozen programs. With the growing need for system optimization, this project hopes to deliver a powerful and extensible replacement for the default task manager to guarantee improved system performance and management.

2. Problem Statement

Current computer users normally face performance problems like CPU hunger, memory exhaustion, and non-responding applications. The Windows Task Manager, though it offers native functionalities for monitoring and process termination, does not support customization, enhanced filtering, and easy-to-use features for in-depth system analysis.

The major problems with the current system are:

1. Limited Customization – The default Task Manager does not permit users to customize the interface or selectively monitor specific system metrics.
2. Lack of Detailed Insights – Although Windows Task Manager displays rudimentary CPU and memory usage, it fails to provide detailed insights into process activity, trends, and historical performance information.
3. Ineffective Process Management – Users are usually unable to easily find unresponsive applications or forcibly kill processes effectively.
4. Complexity for Novices – The Windows Task Manager user interface is daunting for non-technical users who require a simplified yet robust monitoring tool.

To address these issues, this project aims to create a customized task manager with improved functionalities, such as:

1. Real-time CPU usage, memory usage, and running process monitoring.
2. An interactive and customizable user interface based on WPF.
3. Advanced process control functionalities like sorting, filtering, and detailed process descriptions.
4. Enhanced data visualization using graphs and charts to improve system analysis.
5. A small but efficient replacement for Windows Task Manager, which is suitable for technical and non-technical users alike.

This project intends to deliver a real-world solution that not only enhances system monitoring but also improves the user experience by providing more control, personalization, and simplicity. Utilizing the strength of.NET, C#, and WPF, this application delivers effective performance monitoring while having a good-looking and easy-to-use interface.

Chapter 2

Background/ Literature Survey

In today's digital computing age, efficient system resource management is a vital research and development area. With the increased complexity of modern operating systems comes the need for effective tools for monitoring and controlling system performance. Many task management and system monitoring tools have evolved over the years, each with different features and optimizations.

Research and development in this field are aimed at enhancing CPU efficiency, process prioritization, memory management, and real-time monitoring. The Windows Task Manager has been a popular solution, but its shortcomings have prompted the creation of other task managers and system monitoring tools with advanced features. These tools are intended to offer better visualization, greater insight into system performance, and more effective process control mechanisms.

This chapter delves into existing research and developments in task management, process monitoring, and system performance tracking. It also describes some of the most prominent task management solutions today, their advantages and limitations.

2.1 Existing System and Tools

Windows Task Manager is the built-in system monitoring utility that comes preinstalled in every recent Windows operating system. It shows real-time details of CPU utilization, memory usage, currently running applications, and system performance. The primary functions of Windows Task Manager are:

1. Process Management: Enables users to launch, halt, and schedule current processes.
2. Performance Monitoring: Shows CPU, memory, disk, and network usage.
3. Startup Program Management: Assists in managing which programs run at system startup.
4. User Session Monitoring: Displays live user sessions and enables administrators to control them.

There are some limitations in the existing system.

1. Limited customization and filtering options.
2. Basic process descriptions without detailed insights.
3. No advanced visualization features for performance analysis.
4. Lacks deeper system analytics such as historical CPU usage trends.

2.2 Third-Party Task Managers

Due to the limitations of Windows Task Manager, several third-party task management tools have been developed. Some of the most popular alternatives include:

1. **Process Explorer (Sysinternals Suite)**
    a. Provides a more advanced version of Task Manager with deeper insights into processes.
    b. Displays process hierarchies, memory usage, and DLL dependencies.
    c. Limitation: More suited for advanced users; complex interface.

2. Rainmeter

a. A system monitoring tool that provides real-time CPU, RAM, and network statistics in a customizable dashboard.
b. Supports custom skins and widgets for detailed performance tracking.
c. Limitation: Focuses more on system visualization than active process management.

2.3 Research and Development in Task Management

In recent years, research in system performance monitoring has focused on optimizing CPU scheduling, process prioritization, and resource management. Some key areas of research include:

1. **CPU Scheduling and Process Management**

Studies have explored different CPU scheduling algorithms to enhance performance:

    a. Round Robin Scheduling: Ensures equal time allocation to all processes, reducing starvation.
    b. Priority Scheduling: Assigns CPU resources based on process importance.
    c. Multi-Level Queue Scheduling: Separates processes into different priority queues for better management.

2.4 Need for a Custom Task Manager

Based on the limitations of existing solutions and research findings, a custom task manager is required to:

1. Provide a user-friendly interface using WPF and XAML for better visual representation.
2. Offer real-time tracking of CPU and memory usage in an interactive dashboard.
3. Allow advanced filtering and sorting of processes based on user preferences.

The proposed solution aims to combine the efficiency of Windows Task Manager with advanced visualization and filtering options to create a more effective system monitoring tool**.**

Chapter 3

Objectives

The main objectives of the proposed custom task manager are as follows:

1. **Real-Time System Monitoring**
    - To develop a task manager that provides **real-time tracking of CPU usage, memory consumption, and active processes** for better system performance analysis.
2. **Enhanced Process Management**
    - To allow users to **sort, filter, and terminate processes efficiently**, offering better control over system applications compared to the default Windows Task Manager.
3. **User-Friendly and Customizable Interface**
    - To design an **interactive and visually appealing UI** using **WPF and XAML**, ensuring ease of use for both technical and non-technical users.
4. **Advanced Data Visualization**
    - To implement **graphical representations of CPU and memory usage** for better insight into system performance trends and potential bottlenecks.
5. **Lightweight and Efficient Performance**
    - To create an **optimized and resource-efficient application** that does not add excessive load to the system while ensuring smooth and responsive monitoring.

Chapter 4

Hardware and Software Requirements

4.1 Hardware Requirements

S. No	Name of the Hardware	Specification
1	CPU (Processor)	Minimum: intel core i3 /amd Ryzen 3 Recommended: intel core i5,i7 / amd Ryzen 5,7
2	RAM	Minimum: 4 GB Recommended: 8 GB or higher
3	Storage (HDD or SSD)	Minimum: 256 GB HDD Recommended: 512 GB SSD or higher
4	Graphics Card (GPU)	Integrated Graphics (for UI rendering) Recommended: Dedicated GPU for better UI performance
5	Display	Minimum: 1366*768 resolution Recommended: Full HD (1920*1080) or higher

4.2 Software Requirements

S. No	Name of the Software	Specification
1	Operating system	Windows 10/11
2	Development framework	.NET framework (v4.7 or later) /.NET Core
3	Programming language	C# (C- sharp)
4	UI Framework	WPF (Windows presentation Foundation)
5	Markup language	XAML (Extensible Application Markup Language)
6	IDE (Integrated Development Environment)	Visual Studio 2022 or later

Chapter 5

Possible Approach/ Algorithms

Below, we will explore some common scheduling algorithms used in task management systems: **First-Come, First-Served (FCFS)**, **Shortest Job First (SJF)**, **Round-Robin (RR)**, and **Priority Scheduling**. These algorithms help in determining the order in which processes are executed in a system, ensuring efficient CPU time allocation.

1. First-Come, First-Served (FCFS)

Explanation: The **First-Come, First-Served (FCFS)** scheduling algorithm is the simplest and one of the earliest scheduling strategies. It operates on the principle that the process that arrives first will be executed first. In other words, processes are executed in the exact order of their arrival, without any preemption or interruption. FCFS is easy to implement, but it may lead to inefficient CPU utilization, especially if a long process arrives before shorter ones, causing a problem called the **convoy effect**.

Key Points:

- Non-preemptive: Once a process starts executing, it runs to completion.
- Can lead to **high waiting time** if short processes are queued behind long ones.

2. Shortest Job First (SJF)

Explanation: The **Shortest Job First (SJF)** algorithm selects the process with the shortest execution time next. It is designed to minimize the **average waiting time** for a group of processes. SJF can be either **preemptive** or **non-preemptive**. In **preemptive** SJF (also known as **Shortest Remaining Time First (SRTF)**), the process can be preempted if a new process with a shorter burst time arrives. In **non-preemptive** SJF, once a process starts executing, it runs until completion.

Key Points:

- **Non-preemptive** version: The process with the shortest burst time is executed first.
- **Preemptive** version: If a process arrives with a shorter burst time than the current process, it preempts the current process.
- **Ideal for reducing waiting time**, but requires knowing the burst time in advance.

3. Round-Robin Scheduling (RR)

Explanation: The **Round-Robin (RR)** algorithm assigns a fixed time slice or quantum to each process. Each process is allowed to run for a predefined time (time slice) and then is preempted and placed back into the queue, where it waits for its next turn. This ensures that all processes are given a fair share of the CPU and prevents any one process from monopolizing the CPU. RR is commonly used in multi-user and multi-tasking systems.

Key Points:

- **Preemptive**: Processes are preempted after their time slice is completed.
- Ideal for **time-sharing** systems where fairness is important.
- Time quantum needs to be chosen carefully—too large leads to inefficiency, too small results in high overhead.

4. Priority Scheduling

Explanation: In **Priority Scheduling**, each process is assigned a priority, and the process with the highest priority (usually represented by a lower number) is executed first. If two processes have the same priority, the scheduler may use FCFS or another algorithm to decide which one to execute first. Priority Scheduling can be either **preemptive** or **non-preemptive**. In the preemptive version, if a new process arrives with a higher priority, it can preempt the currently running process.

Key Points:

- **Preemptive**: The scheduler can preempt the currently running process if a higher-priority process arrives.
- **Non-preemptive**: Once a process starts, it runs to completion.
- Can lead to **starvation**, where low-priority processes may never get executed if higher-priority ones keep arriving.

Chapter 6

Extra Features

In order to provide a more secure and user-friendly experience, the Task Manager application integrates advanced extra features beyond the core functionalities of task creation, editing, and tracking. These features — App Lock and Voice Recognition — are designed to enhance user privacy, improve accessibility, and simplify interaction with the application. They are especially valuable in real-world scenarios where data security and quick task access are critical for productivity and efficiency.

1. App Lock:
Since the application stores personal tasks, schedules, and reminders — often containing sensitive or private information — it is essential to ensure that unauthorized users cannot access the app content. The App Lock feature adds an extra layer of protection by enabling the user to lock the application using a PIN, password, or biometric authentication such as fingerprint or face recognition.

This security feature activates when the app is launched or resumed after a period of inactivity. It helps prevent accidental or unauthorized access to tasks and reminders, which is especially important for users who use shared devices or mobile phones. The App Lock is lightweight, user-configurable, and integrated directly into the system settings of the application. Users can choose their preferred authentication method and can enable or disable this feature at any time. By incorporating this feature, the Task Manager not only ensures data privacy but also builds trust and reliability among users.

2. Voice Recognition:
Voice Recognition is included in the Task Manager to facilitate hands-free interaction and improve accessibility. With this feature, users can create, modify, delete, or retrieve tasks simply by using their voice. This is particularly helpful in situations where typing is inconvenient — such as when the user is driving, multitasking, or has limited mobility.

The system uses the device's built-in voice recognition capabilities to process spoken commands such as "Add a new task," "What's on my to-do list?", or "Mark meeting as complete." Once the voice input is received, it is converted to text and matched with specific actions within the app. The interface provides audible or visual confirmation to ensure users know their commands were understood and executed.

This feature increases efficiency and allows users to manage tasks quickly and naturally, without the need to navigate multiple menus. It also supports accessibility for users with visual or motor impairments, making the Task Manager more inclusive and adaptive to diverse user needs.

By implementing App Lock and Voice Recognition, the Task Manager application goes beyond the traditional task tracking system and provides a secure, intelligent, and convenient user experience. These features align with modern trends in application development and contribute significantly to user satisfaction and productivity.

References

[1] A. Silberschatz, P. B. Galvin, and G. Gagne, "Operating System Concepts," 9th ed., John Wiley & Sons, 2012. (Example: Textbook)

[2] W. Stallings, "Operating Systems: Internals and Design Principles," 8th ed., Pearson Education, 2014. (Example: Textbook)

[3] H. M. Deitel, P. J. Deitel, and D. R. Choffnes, "Operating Systems," 3rd ed., Pearson, 2003. (Example: Textbook)

[4] R. Buyya, T. S. Dillon, and J. G. Martinez, "Cloud Computing: Principles and Paradigms," Wiley, 2011. (Example: Textbook)

[5] P. J. Denning, "The working set model for program behavior," Commun. ACM, vol. 11, no. 5, pp. 323-333, 1968. (Example: Journal Paper)