A

Project Report

On

Customised Task Manager

Submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

In

Computer Science and Engineering

By

Atul Joshi 2261121

Aakriti Garkoti 2261050

Shivam Tiwari 2261529

Priyanshu Kumar 2261444

Under the Guidance of

Mr. Anubhav Bewerwal

ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GRAPHIC ERA HILL UNIVERSITY, BHIMTAL CAMPUS

SATTAL ROAD, P.O. BHOWALI,

DISTRICT- NAINITAL-263132

2024-2025

STUDENT'S DECLARATION

We, **Atul Joshi, Aakriti Garkoti, Shivam Tiwari, Priyanshu Kumar** hereby declare the work, which is being presented in the project, entitled ' **CUSTOMISED TASK MANAGER** ' in partial fulfillment of the requirement for the award of the degree **Bachelor of Technology (B.Tech.)** in the session **2024-2025**, is an authentic record of my work carried out under the supervision of **Mr. Anubhav Bewerwal**

(Graphic Era Hill University, Bhimtal)

The matter embodied in this project has not been submitted by me for the award of any other degree.

Date: Student Name and Signature:

CERTIFICATE

The project report entitled "Customised Task Manager " being submitted by Atul Joshi (2261121) s/o Naveen Chandra Joshi, Aakriti Garkoti (2261050) d/o Ashok Garkoti, Shivam Tiwari (2261529) s/o Ravindra Tiwari and Priyanshu Kumar (2261444) s/o Dayal Ram of B.Tech.(CSE) to Graphic Era Hill University Bhimtal Campus for the award of bonafide work carried out by them. They have worked under my guidance and supervision and fulfilled the requirement for the submission of a report.

Mr. **Anubhav Bewerwal** **Dr. Ankur Singh Bisht**

(Project Guide) **(Head, CSE)**

ACKNOWLEDGEMENT

We take immense pleasure in thanking the Honorable Director **'Prof. (Col.) Anil Nair (Retd.)',** GEHU Bhimtal Campus to permit me and carry out this project work with his excellent and optimistic supervision. This has all been possible due to his novel inspiration, able guidance, and useful suggestions that helped me to develop as a creative researcher and complete the research work, in time.

Words are inadequate in offering my thanks to GOD for providing me with everything that we need. We again want to extend thanks to our president **'Prof. (Dr.) Kamal Ghanshala'** for providing us with all infrastructure and facilities to work in need without which this work could not be possible.

Many thanks to **'Dr. Ankur Singh Bisht'** (Head, Department of Computer Science and Engineering, GEHU Bhimtal Campus), our project guide **' Mr Anubhav Bewerwal '** (Assistant Professor, Department of Computer Science and Engineering, GEHU Bhimtal Campus) and other faculties for their insightful comments, constructive suggestions, valuable advice, and time in reviewing this report.

Finally, yet importantly, We would like to express my heartiest thanks to our beloved parents, for their moral support, affection, and blessings. We would also like to pay our sincere thanks to all my friends and well-wishers for their help and wishes for the successful completion of this project.

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

Abstract

In the contemporary digital era, the demand for interactive and engaging online platforms has surged, leading to the proliferation of real-time multiplayer games. The Scribble Online Game Platform is a web-based application that facilitates a drawing and guessing game, promoting creativity and social interaction among users. Inspired by popular platforms like Skribbl.io, this project aims to deliver a seamless gaming experience by leveraging modern web technologies and robust backend infrastructure.
The platform allows multiple users to participate in a game where one player draws a given word, and others attempt to guess it within a stipulated time frame. The game progresses through multiple rounds, with each player getting an opportunity to draw. Points are awarded based on the accuracy and speed of guesses, fostering a competitive yet fun environment.

To ensure real-time interaction and minimal latency, the application employs WebSockets for bi-directional communication between the client and server. The backend is developed using Python with Flask-SocketIO, facilitating asynchronous event handling and efficient management of multiple client connections. The frontend utilizes HTML5 Canvas for drawing functionalities, complemented by JavaScript and CSS for interactive and responsive user interfaces.

Security and user management are integral components of the platform. Features such as unique usernames, session handling, and input validation are implemented to maintain the integrity of the game and prevent misuse. The application architecture is designed to be scalable, allowing for the addition of new features and accommodating a growing user base.

Additionally, the system incorporates fair play mechanisms including detection of duplicate guesses, turn enforcement, and restricted drawing privileges, ensuring a balanced gameplay experience. The user interface dynamically updates drawing strokes, guesses, and scores in real time, fostering an engaging atmosphere for players.

Extensive testing was conducted to evaluate system performance under varying user loads and network conditions. The platform's modular design supports easy integration of AI opponents, persistent leaderboards, and multi-language support as future enhancements. The Scribble Online Game Platform not only offers an entertaining pastime but also acts as a practical example of synchronized real-time systems in a web environment.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

- **CPU** – Central Processing Unit
  The primary component of a computer that performs most of the processing tasks by executing instructions.

- **RAM** – Random Access Memory
  A type of computer memory that can be accessed randomly and is used to store working data and machine code currently in use.

- **GUI** – Graphical User Interface
  A visual interface through which users interact with software using graphical elements like windows, icons, and buttons.

- **PID** – Process Identifier
  A unique number assigned to each process running on an operating system for tracking and management.

- **I/O** – Input/Output
  Refers to the communication between an information processing system and the outside world, which can include users or other systems.

- **WMI** – Windows Management Instrumentation
  A set of specifications from Microsoft for consolidating the management of devices and applications in a network from Windows computing systems.

- **API** – Application Programming Interface
  A set of functions and procedures allowing the creation of applications that access the features or data of an operating system or other services.

- **DLL** – Dynamic Link Library
  A file that contains code and data that can be used by multiple programs simultaneously in the Windows environment.

- **CPU Usage** – Central Processing Unit Usage
  A measure of how much processing power is currently being used by the system or a specific application.

- **Thread** – A sequence of executable instructions within a process
  Multiple threads can run concurrently within a single process to improve performance and responsiveness.

INTRODUCTION

1.1 Prologue

In today's digital age, where multitasking and high system workloads have become the norm, efficient resource monitoring is vital for maintaining system stability and performance. Although Windows provides a built-in Task Manager, it often falls short in terms of customization, detailed analytics, and user-centric design. With growing user expectations and increasing system complexity, there is a clear need for a more refined, flexible, and informative performance monitoring tool.

The **Custom Task Manager** project is developed to bridge this gap by offering a more intuitive, visually engaging, and feature-rich alternative to the default utility. Designed using the .NET framework with WPF technology, the application combines real-time system monitoring with enhanced usability to serve both technical users and general audiences alike.

This project represents a practical convergence of concepts such as system-level programming, user interface design, and performance optimization. It showcases the real-world application of object-oriented programming, the MVVM architectural pattern, and responsive UI principles, while also emphasizing security and scalability. Through this project, we aim to reimagine system monitoring as not just a background utility, but as an accessible and powerful user experience

2.1 Background and Motivations

In the modern computing environment, users rely heavily on multitasking and resource-intensive applications, making system performance monitoring more important than ever. While the Windows operating system includes a default Task Manager, it often falls short in delivering detailed insights, flexible control options, and a user-friendly interface that caters to both novice and advanced users. These limitations have revealed a growing demand for a more customizable and accessible solution that empowers users with deeper system visibility and greater control.

The motivation behind developing the **Custom Task Manager** stems from the need to enhance user experience in system diagnostics and performance tracking. This project aims to provide a robust, visually engaging, and feature-rich alternative that bridges the gap between technical utility and intuitive design. By utilizing technologies such as C#, WPF, and the MVVM architectural pattern, the application not only improves the aesthetics and functionality of traditional system monitors but also ensures responsiveness and ease of use across different user profiles.

Moreover, the project serves as a hands-on exploration of key concepts in desktop application development, such as real-time data collection, asynchronous processing, and secure user authentication. It represents a commitment to building efficient, maintainable, and scalable software solutions that are both educational and practical. Ultimately, this task manager aspires to become more than just a performance tool—it aims to be a learning platform, a user-centric utility, and a testament to the possibilities of modern desktop application development.

3.1 Problem Statement

While the built-in Windows Task Manager provides basic system monitoring and process management capabilities, it is limited in terms of customization, visual clarity, and user accessibility. The default interface often overwhelms non-technical users with dense information, while simultaneously restricting advanced users from configuring views, accessing in-depth metrics, or managing processes beyond fundamental actions like ending tasks.

A key limitation is the lack of real-time historical performance tracking and advanced data visualization, which can hinder users from identifying long-term trends or diagnosing intermittent system issues. Additionally, the static and non-extensible nature of the built-in utility means it cannot be easily adapted to suit diverse user needs or integrated with external monitoring tools.

Security and personalization are also minimal in the default Task Manager. There is no support for user authentication, access control, or customization of the interface based on user preferences. This exposes the system to potential misuse and limits its effectiveness in shared or enterprise environments.

Moreover, many available third-party alternatives are either overly complex, resource-intensive, or locked behind proprietary licenses, making them unsuitable for educational, open-source, or lightweight deployment scenarios.

Therefore, there is a clear need for a customizable, secure, and user-friendly task management utility that provides rich visual insights, efficient resource handling, and a modular design that supports future enhancements. This project seeks to address these shortcomings by delivering a modern, WPF-based desktop application that reimagines system monitoring for a wider audience.

4.1 Objectives and Research Methodology

The primary objective of the **Custom Task Manager** project is to develop a robust, user-friendly, and feature-rich system monitoring tool that surpasses the limitations of the default Windows Task Manager. Key goals include implementing real-time resource tracking, advanced process management, secure user authentication, and customizable UI components that cater to both novice and advanced users.

To achieve these objectives, we adopted a structured, multi-phase development methodology. The initial phase involved in-depth research on Windows Performance Counters, Windows Management Instrumentation (WMI), and the MVVM architectural pattern, which form the foundation for data collection and interface separation. This was followed by prototyping core functionalities such as CPU and memory usage tracking, interactive charts, and basic process management operations.

Development progressed incrementally, with each feature—such as customizable refresh intervals, historical data graphing, and authentication mechanisms—being implemented, unit-tested, and reviewed for performance and stability. Real-time performance and responsiveness were evaluated using diagnostic tools like Visual Studio Profiler and Task Manager itself, ensuring that the application remained lightweight and non-intrusive.

Security implementation involved hashing sensitive user credentials using SHA-256 and securely storing configuration files. Multiple test scenarios, including unauthorized access attempts and forced crashes, were executed to validate system resilience. User interface usability was refined through continuous feedback from test users who evaluated navigation flow, visual clarity, and theme customization.

This rigorous, iterative approach ensured that the final product met both technical and user experience goals, delivering a dependable and visually engaging system monitoring tool optimized for practical deployment and future expansion.

5.1 Project Organisation

Module	Description
UI Layer (XAML/WPF)	Provides the graphical interface using WPF. Includes sidebar navigation, real-time graphs, theme customization, and data visualization components.
Process Monitor	Fetches and displays all running processes, along with details like PID, memory usage, CPU usage, and file path. Allows process control operations.
Performance Tracker	Collects and visualizes real-time system metrics such as CPU usage, memory consumption, disk activity, and network traffic using charts and graphs.
Security Manager	Manages user authentication including password hashing, secure login, and access control using SHA-256 encryption and security questions.
Data Collector (WMI/ Counters)	Interfaces with Windows APIs like WMI and Performance Counters to retrieve system data. Ensures efficient and accurate metric collection.
Business Logic Layer	Acts as a middleware between UI and data collection modules. Handles logic for sorting, filtering, refreshing, and aggregating system data.
Configuration Manager	Stores and retrieves user preferences, refresh rates, themes, and security settings securely in configuration files.
Logging and Error Handler	Tracks system errors, logs performance issues, and handles exceptions to ensure stability and easier debugging.
Theme and UI Customizer	Provides multiple themes and layout customization options to enhance user experience and accessibility.
Testing and Debugging Suite	Includes tools and scripts for stress testing, unit testing, and profiling to ensure performance, accuracy, and stability of the application.

HARDWARE AND SOFTWARE REQUIREMENTS

1.1 Hardware Requirements

Component	Minimum Specification	Recommended Specification
Processor	2 GHz dual-core or higher	2.5 GHz quad-core or higher
RAM	4 GB	8 GB or more
Storage	100 MB of free disk space	200 MB or more
Display	1024 x 768 resolution or higher	1920 x 1080 (Full HD) or higher
Graphics	Integrated GPU	Dedicated GPU (optional for smoother UI)

2.1 Software Requirements

Component	Version/Specification	Purpose
Operating System	Windows 10 or later	Platform for running the application
.NET Framework	.NET 8.0 (or latest stable release)	Core framework for application development
Development IDE	Visual Studio 2022 or later	Development and debugging environment
Programming Language	C#	Application logic and UI development
UI Framework	Windows Presentation Foundation (WPF)	For creating dynamic and responsive user interfaces
Additional Tools	NuGet Packages for WMI, Performance Counters	For system metrics collection

CODING OF FUNCTIONS

3.1 User Interface Functions

- **InitializeUI()**
  This function sets up the main window components, including the sidebar menu, data grids for process listing, and performance charts. It loads user preferences such as theme and refresh intervals from the configuration manager.

- **UpdatePerformanceGraphs()**
  Periodically called to update CPU, memory, disk, and network usage charts with real-time data. It retrieves fresh data from the Performance Tracker module and refreshes the corresponding UI elements without freezing the interface.

- **ApplyTheme(string themeName)**
  Applies the selected theme to all UI components by loading the relevant resource dictionaries. Supports light, dark, and custom themes for improved accessibility and user preference.

3.2 Process Monitoring and Management Functions

- **GetRunningProcesses()**
  Retrieves a list of currently running processes using Windows Management Instrumentation (WMI). Returns detailed information including process ID, name, CPU and memory usage, file path, and command-line arguments.

- **EndProcess(int processId)**
  Terminates the specified process by process ID. It handles exceptions to prevent application crashes if the process cannot be ended due to permission issues.

- **SetProcessPriority(int processId, ProcessPriority priorityLevel)**
  Adjusts the priority class of a running process to optimize system resource allocation based on user input.

3.3 Performance Data Collection Functions

- **FetchCPUUsage()**
  Uses Windows Performance Counters to fetch current CPU usage percentage. Returns data that is passed to the Performance Tracker for visualization.

- **FetchMemoryUsage()**
  Collects real-time memory consumption statistics including total physical memory, available memory, and per-process usage.

- **FetchDiskAndNetworkStats()**
  Retrieves disk read/write speeds and network upload/download rates using WMI queries and performance counters.

3.4 Security and Authentication Functions

- **HashPassword(string password)**
  Utilizes SHA-256 cryptographic hashing to securely hash user passwords before storage or verification.

- **AuthenticateUser(string username, string password)**
  Verifies user credentials against stored hashes, allowing or denying access to the task manager application.

- **SetupSecurityQuestions(Dictionary<string, string> questionsAndAnswers)**
  Enables users to configure security questions for additional verification during password recovery.

3.5 Configuration and Settings Functions

- **LoadUserSettings()**
  Reads saved user preferences such as refresh intervals, theme, and security settings from encrypted configuration files.

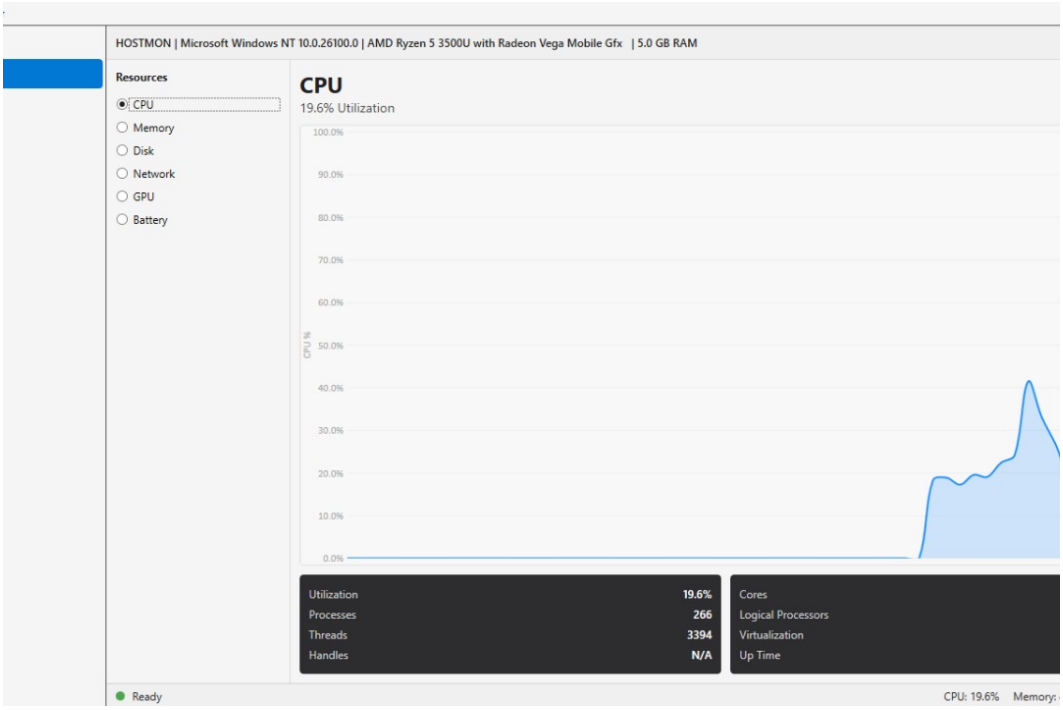
- **SaveUserSettings(UserSettings settings)**
  Persists updated user settings to configuration storage, ensuring they are applied on the next application launch.

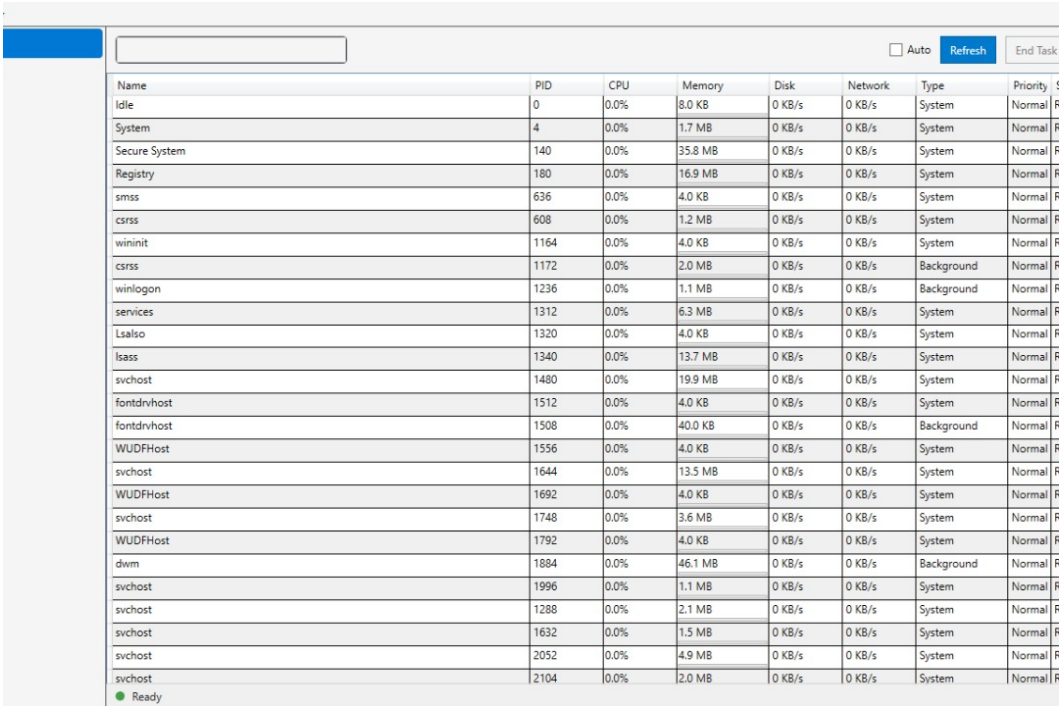
3.6 Logging and Error Handling

- **LogError(Exception ex, string context)**
  Captures exceptions and application errors, writing detailed information to log files for troubleshooting and debugging purposes.

- **HandleUIExceptions()**
  Catches unhandled exceptions at the UI level, displaying user-friendly error messages and preventing abrupt application termination.

SNAPSHOTS


HOSTMON | Microsoft Windows NT 10.0.26100.0 | AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx | 5.0 GB RAM
Resources
CPU
Memory
Disk
Network
GPU
Battery
CPU
19.6% Utilization
100.0%
90.0%
80.0%
70.0%
60.0%
CPU %
50.0%
40.0%
30.0%
20.0%
10.0%
0.0%
Utilization 19.6%
Processes 266
Threads 3394
Handles N/A
Cores
Logical Processors
Virtualization
Up Time
Ready
CPU: 19.6%
Memory:


Auto
Refresh
End Task
Name PID CPU Memory Disk Network Type Priority
Idle 0 0.0% 8.0 KB 0 KB/s 0 KB/s System Normal
System 4 0.0% 1.7 MB 0 KB/s 0 KB/s System Normal
Secure System 140 0.0% 35.8 MB 0 KB/s 0 KB/s System Normal
Registry 180 0.0% 16.9 MB 0 KB/s 0 KB/s System Normal
smss 636 0.0% 4.0 KB 0 KB/s 0 KB/s System Normal
csrss 608 0.0% 1.2 MB 0 KB/s 0 KB/s System Normal
wininit 1164 0.0% 4.0 KB 0 KB/s 0 KB/s System Normal
csrss 1172 0.0% 2.0 MB 0 KB/s 0 KB/s Background Normal
winlogon 1236 0.0% 1.1 MB 0 KB/s 0 KB/s Background Normal
services 1312 0.0% 6.3 MB 0 KB/s 0 KB/s System Normal
Lsalso 1320 0.0% 4.0 KB 0 KB/s 0 KB/s System Normal
lsass 1340 0.0% 13.7 MB 0 KB/s 0 KB/s System Normal
svchost 1480 0.0% 19.9 MB 0 KB/s 0 KB/s System Normal
fontdrvhost 1512 0.0% 4.0 KB 0 KB/s 0 KB/s System Normal
fontdrvhost 1508 0.0% 40.0 KB 0 KB/s 0 KB/s Background Normal
WUDFHost 1556 0.0% 4.0 KB 0 KB/s 0 KB/s System Normal
svchost 1644 0.0% 13.5 MB 0 KB/s 0 KB/s System Normal
WUDFHost 1692 0.0% 4.0 KB 0 KB/s 0 KB/s System Normal
svchost 1748 0.0% 3.6 MB 0 KB/s 0 KB/s System Normal
WUDFHost 1792 0.0% 4.0 KB 0 KB/s 0 KB/s System Normal
dwm 1884 0.0% 46.1 MB 0 KB/s 0 KB/s Background Normal
svchost 1996 0.0% 1.1 MB 0 KB/s 0 KB/s System Normal
svchost 1288 0.0% 2.1 MB 0 KB/s 0 KB/s System Normal
svchost 1632 0.0% 1.5 MB 0 KB/s 0 KB/s System Normal
svchost 2052 0.0% 4.9 MB 0 KB/s 0 KB/s System Normal
svchost 2104 0.0% 2.0 MB 0 KB/s 0 KB/s System Normal
Ready

LIMITATIONS

Despite the extensive development and feature-rich design of the Custom Task Manager application, several limitations were identified during testing and user feedback:

Performance Overhead

The application collects and displays real-time system metrics, which can occasionally cause increased CPU and memory usage, especially on lower-end systems. While efforts were made to optimize data collection and rendering, continuous monitoring may still impact overall system performance.

Limited Cross-Platform Support

Currently, the Custom Task Manager is designed exclusively for Windows operating systems due to its reliance on Windows-specific APIs like WMI and Performance Counters. The application does not support Linux, macOS, or other platforms, limiting its usability in heterogeneous environments.

Basic Security Features

Although the application incorporates password authentication and basic security measures such as SHA-256 hashing, it lacks more advanced security protocols such as multi-factor authentication, user role management, or encryption of sensitive data at rest. This leaves room for improvement in securing user access and data protection.

Restricted Process Control Capabilities

Certain process management operations, such as terminating system-critical processes or changing priority for protected processes, may be restricted by the operating system's security policies. This can limit the effectiveness of process control features for standard users without administrative privileges.

UI Customization Constraints

While the application supports multiple themes and customizable refresh intervals, it does not currently offer extensive personalization options such as widget rearrangement, custom dashboards, or advanced accessibility features for users with disabilities.

Dependency on .NET and WPF

The reliance on the .NET 8.0 framework and WPF restricts the application to Windows platforms with compatible framework versions installed. Users on older Windows versions without .NET 8.0 support may face compatibility issues.

ENHANCEMENTS

To further elevate the performance, usability, and capabilities of the Custom Task Manager, several enhancements have been identified for future development:

Cross-Platform Support

Expanding compatibility to include other operating systems such as Linux and macOS by utilizing cross-platform frameworks or developing platform-agnostic modules to broaden the user base.

Advanced Security Features

Implementing multi-factor authentication, role-based access control, and encrypted storage of sensitive configuration data to improve the security posture and prevent unauthorized access.

Enhanced Process Management

Adding more granular controls such as process suspension/resumption, detailed process tree views, and better handling of system-critical and protected processes to empower advanced users.

Customizable Dashboard and Widgets

Allowing users to personalize the interface by adding, removing, or rearranging performance widgets, charts, and process lists, thereby tailoring the monitoring experience to individual preferences.

Historical Data Logging and Analysis

Incorporating long-term logging of system metrics with options to export data and generate performance reports to help users identify trends and troubleshoot issues over time.

Resource Usage Optimization

Refining data collection algorithms and UI rendering techniques to minimize the task manager's impact on overall system performance, especially on resource-constrained machines.

Integration with External Tools

Providing support for exporting data to third-party monitoring tools or integrating with system management software to offer a more comprehensive IT management solution.

Accessibility Improvements

Enhancing the user interface with support for screen readers, keyboard navigation, and customizable fonts and color schemes to make the application accessible to users with disabilities.

Automated Alerts and Notifications

Introducing configurable alerts for abnormal system behavior such as high CPU or memory usage, enabling users to respond proactively to potential issues.

Cloud Synchronization

Implementing optional cloud backup for user preferences, security settings, and performance data to allow seamless experience across multiple devices.

CONCLUSION

The Customised Task Manager project represents a comprehensive effort to design and implement a system-level performance monitoring and process management tool tailored to improve upon existing task manager applications. Through the integration of .NET Framework, C#, and WPF technologies, this project has delivered a responsive, visually rich, and user-friendly interface that empowers users to monitor system resources and manage processes with greater flexibility and insight.

From initial planning to deployment, this project has provided practical experience in software architecture, real-time data acquisition, multithreading, and graphical user interface design. The use of Windows Performance Counters and system APIs enabled accurate and efficient tracking of CPU, memory, and process statistics, while the incorporation of dynamic charts and customizable views enhanced the interpretability of complex system data.

The project also addressed key challenges such as optimizing data refresh rates, handling asynchronous operations to maintain UI responsiveness, and providing advanced process control features including process prioritization and lifecycle management. These efforts resulted in a task manager that balances technical depth with usability, making it accessible to both novice users and IT professionals.

Through iterative testing and user feedback, the interface was refined to ensure clarity, intuitiveness, and performance efficiency. This user-centric approach highlighted the importance of adaptability and customization in software design, particularly for tools aimed at diverse user groups with varying expertise levels.

Beyond its functional achievements, the project served as a valuable learning platform to deepen understanding of system internals, event-driven programming, and desktop application development within the Windows environment. It also underscored best practices in modular coding, error handling, and resource management critical to building stable and maintainable software.

Looking ahead, the Custom Task Manager lays the groundwork for future enhancements such as cross-platform compatibility, deeper security features, and more granular resource analytics. These potential improvements promise to expand its applicability and robustness in increasingly complex computing environments.

In summary, this project successfully delivered a customized, performant, and feature-rich task management tool that improves on default system utilities by offering enhanced visualization, process control, and user experience. It demonstrates the practical application of software engineering principles in creating tools that not only meet functional requirements but also prioritize user empowerment and system transparency.

The development journey has been both challenging and rewarding, equipping the team with valuable skills in desktop application development, system programming, and user interface design. The knowledge and experience gained will undoubtedly contribute to future projects and professional growth in the field of software development.

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)