Chapter 1: Introduction

1.1 Project Summary

TaskCore is a Windows desktop utility developed in C# and WPF, designed to function as an advanced process manager. It is a lightweight and user-friendly tool that monitors running processes in real-time, offers accurate CPU and memory usage data, and allows users to terminate tasks directly from its interface. The application is built with a modular and extensible architecture, making it easy to add new features in the future.

1.2 Project Purpose

The primary purpose of the project is to provide users with a modern, simple, and interactive interface that is cleaner than the standard Windows Task Manager. It aims to offer real-time visibility into all active processes and facilitate easy management of tasks, such as terminating unresponsive or resource-intensive applications.

1.3 Project Scope

The scope of the TaskCore project includes the development of a fully functional task manager with several core and advanced features. The application will be structured with a tab view, navigated by a hamburger menu, to switch between different functionalities like the process list and system graphs.

- **Process List**: A comprehensive list of all running tasks displaying their Name, PID, CPU usage (%), and Memory usage (MB). The list is presented in a tree view that can be expanded or collapsed.
- **Task Termination**: An "End Task" button is available to terminate any selected process.
- **Auto-Refresh**: The process list automatically refreshes every second to provide real-time data.
- **System Graphs**: The application displays graphs for CPU and memory usage over time.
- **Watchdog**: An advanced feature that provides alerts if a banned task is found running.

1.4 Objectives

1.4.1 Main Objective

- To provide users with real-time visibility into all active processes.
- To allow for the easy management and termination of tasks.
- To deliver accurate CPU and memory usage statistics with a periodic refresh rate.

- To ensure the application runs with the necessary administrator permissions to manage system tasks.

1.4.2 Secondary Objectives

- To build an extensible and modular architecture that simplifies the process of adding new features.
- To design a modern, clean, and interactive user interface.
- To implement advanced features, including performance graphs, a RAM cleanup utility, and a process watchdog for alerts

Chapter 2: Literature Review

2.1 Introduction of Survey

A literature survey was conducted to analyze the existing landscape of system monitoring tools and to evaluate the technologies best suited for the development of the TaskCore application. This review covers an analysis of the primary existing system (Windows Task Manager), the selection of the core software development technologies, and the architectural patterns chosen to meet the project's objectives. The goal is to leverage established technologies to build an improved and more user-friendly utility.

2.2 Why Survey?

The survey was essential to identify functional gaps and user experience limitations in currently available tools, justifying the need for a new application. While the default Windows Task Manager is a comprehensive utility, its interface can be overwhelming for some users. The purpose of TaskCore is to offer a cleaner, more intuitive, and modern alternative that focuses on core functionalities while reducing complexity. This project aims to provide a lightweight, focused tool that adds value through advanced features like integrated performance graphs and a process watchdog, which are not presented in the same way in the standard manager.

2.3 Comparative analysis with existing systems

Feature	Windows Task Manager	TaskCore
User Interface (UI)	A multi-tabbed interface that separates processes, performance, and services. The design is information-dense and can be complex for novice users.	A modern, single-view UI with a clean, dark theme and a collapsible sidebar for navigation. The design prioritizes simplicity and user experience.
Process Display	Displays processes in a flat list by default. While grouping is possible, it is not the primary mode of display.	Automatically groups processes with the same name into an expandable tree view. This provides a cleaner look and shows aggregate resource usage for the group.
Data Refresh Rate	The update speed is configurable but is not prominently set to a high-frequency real-time speed by default.	The process list is designed to auto-refresh every 1 second, ensuring data is presented in near real-time.
Performance Graphs	Performance graphs are detailed but are located in a separate "Performance" tab, detached from the main process list.	System graphs for CPU and memory are integrated into their own panel, accessible directly from the main sidebar, providing a more cohesive experience.
Custom Features	Does not include specialized features like automatically terminating a predefined list of applications.	Includes a unique "Watchdog" feature that allows a user to create a blocklist of applications and have TaskCore automatically terminate them if they are found running.

Chapter 3: Project Management

3.1 Project Planning Objectives

This chapter outlines the project management approach used for the development of the TaskCore application, including planning objectives, resource allocation, scheduling, and risk management strategies.

3.1.1 Software Scope

The scope of the project is to deliver a fully functional desktop application with the following core features:

- A process list displaying the name, PID, CPU usage, and memory usage of all running tasks.
- A tree view for grouping multiple instances of the same process.
- An "End Task" button for immediate process termination.
- An auto-refresh mechanism that updates the process list every second.
- An integrated panel for displaying real-time system performance graphs for CPU and memory.
- A "Watchdog" feature to automatically terminate user-defined banned processes.

3.1.2 Resources

3.1.2.1 Human Resources

The project was carried out by Joshi Rishi with the supervision of Ms. Anjali Chopra

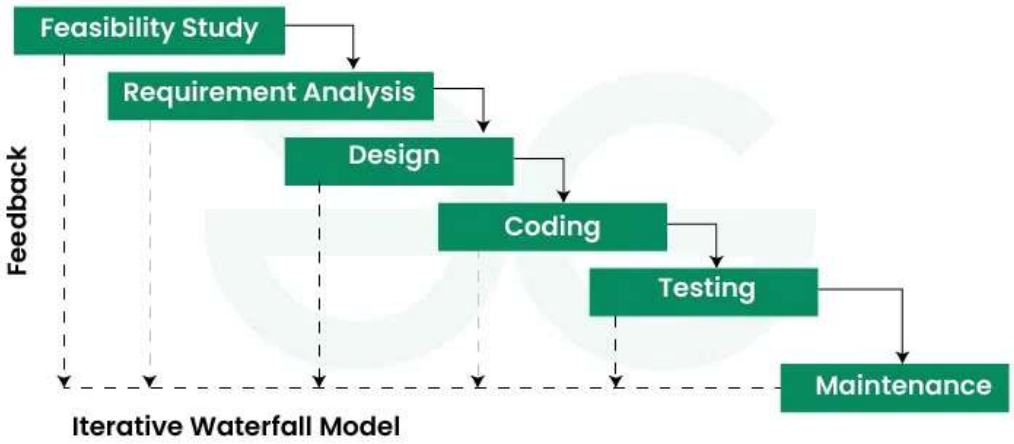
3.1.2.2 Reusable Software Resources

The project leverages the .NET Framework and its extensive class libraries, particularly the Windows Presentation Foundation (WPF) for the user interface and the System.Diagnostics namespace for process management.

3.1.2.3 Environmental Resources:

Development and testing were conducted in a Windows environment using standard development tools like a .NET Integrated Development Environment (IDE). The target environment for deployment is modern Windows operating systems.

3.1.3 Project Development Approach



The TaskCore project adopts the Iterative Waterfall Model as the primary development methodology. This approach combines the structured phases of the traditional waterfall model with the flexibility of iterative development.

- **Iterative Waterfall Model**

  The model was selected for its systematic approach and ability to accommodate refinements during development. The project follows five sequential phases:

  - **Requirements Analysis -** Identification of system requirements and functional specifications
  - **System Design -** Architecture design and technical specifications for all modules
  - **Implementation -** Sequential development of modules using WPF/C# technologies
  - **Testing -** Unit testing, integration testing, and system validation
  - **Deployment -** Final packaging and documentation

- Key Benefits

  - **Structured Development:** Clear phases with defined milestones
  - **Quality Control:** Review checkpoints at each phase
  - **Flexibility:** Allows iterative improvements while maintaining structure
  - **Documentation:** Comprehensive documentation maintained throughout development

3.2 Project Scheduling

3.2.1 Basic Principle: The core principle of scheduling was to deconstruct the complex application into a series of well-defined tasks that could be completed and tested incrementally.

3.2.2 Compartmentalization: The project was naturally compartmentalized into its main feature modules: the core application, the process list, the graphs panel, and the watchdog panel. This separation allowed for parallel planning and streamlined development.

3.2.3 Work Breakdown Structure (WBS)

The project was broken down into a Work Breakdown Structure (WBS) to provide a clear, hierarchical decomposition of all the work necessary to complete the project. This structure is essential for accurate planning, scheduling, and task assignment. The diagram below illustrates the WBS for the TaskCore project.

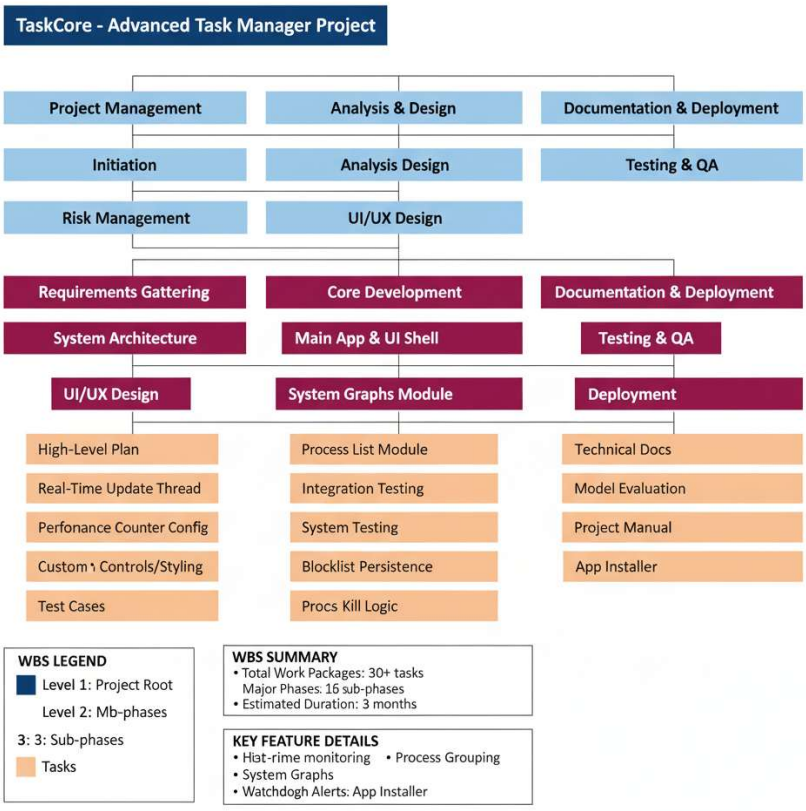


Figure 3.1: Work Breanddan Structure (WBS) for the TaskCore Project

As shown in Figure 3.1, the project is divided into five top-level phases: Project Management, Analysis & Design, Core Development, Testing & QA, and Documentation & Deployment. The Core Development phase is further broken down into the key functional modules of the application: the Main App & UI Shell, the System Graphs Module, and the Watchdog Module. This hierarchical structure ensures that each feature is developed as a self-contained unit, which aligns with the project's modular and iterative development approach and simplifies the testing process.

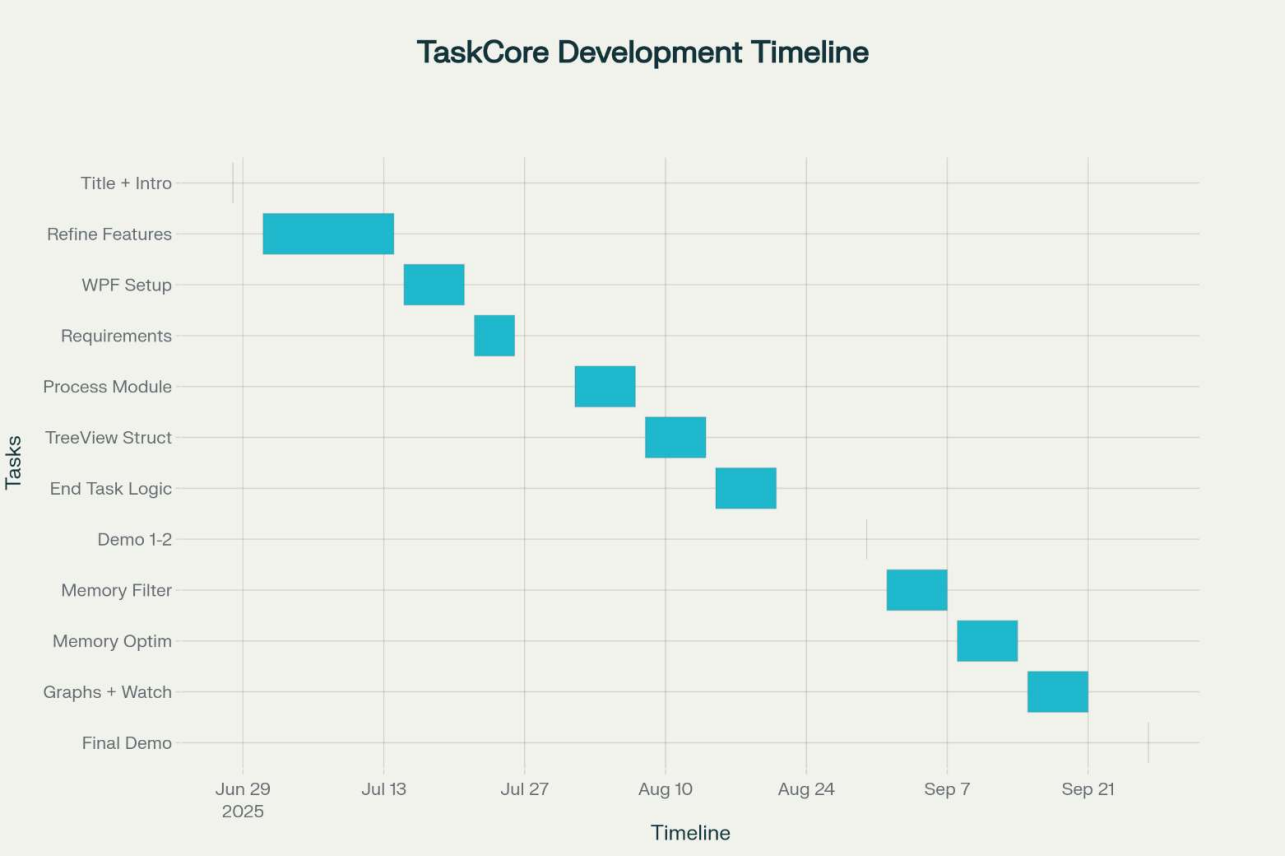
3.2.4 Project Organization

The TaskCore project was undertaken and developed by a single developer, Joshi Rishi, under the expert guidance of Ms. Anjali Chopra. This structure ensured a focused and streamlined development process, combining direct execution with experienced oversight.

The roles and responsibilities were defined as follows:

- **Developer (Joshi Rishi):** As the sole developer, Joshi Rishi was responsible for the entire project lifecycle. This encompassed all phases, including the initial system analysis and design, the development and coding of all core modules like the Process List and Watchdog, and conducting all quality assurance activities, such as black-box and white-box testing.
- **Project Guide (Ms. Anjali Chopra):** Ms. Anjali Chopra served as the project guide, providing mentorship and high-level supervision. Her role was to offer direction, review the project's progress, and ensure that the development work remained aligned with the stated objectives.

3.2.5 TimeLine Chart



3.2.5.1 Time Allocation

MONTH	WEEK	WORK DESCRIPTION	DELIVERABLES
JUNE	Week 4 (28 June)	Final Title Submission + Intro Presentation	Project approval
JULY	Week 1-2	Refine feature list, finalize modules, sketch UI (hamburger layout)	Feature specification
	Week 3	Begin WPF project setup, implement base UI and hamburger menu layout	Base application framework
	Week 4 (26 July)	Requirement Analysis + Final Workflow Submission	Requirements document
AUGUST	Week 1	Build process list module (Name, PID, RAM, CPU, Path) using Process API + WMI	Process monitoring module
	Week 2	Add Tree View structure for parent/subprocess grouping with expand/collapse	Hierarchical process view
	Week 3	Implement End Task + Admin check logic	Process termination functionality
	Week 4 (30 Aug)	Demo 1-2 Modules- Show: process list, Tree View, End Task working	Mid-project demonstration
SEPTEMBER	Week 1	Add memory filter + auto-refresh logic (1s values, 5s structure reload)	Real-time monitoring
	Week 2	Optimize memory usage: cache icons, reduce load, cap under 200MB	Performance optimization
	Week 3	Add graphs tab (LiveCharts2 for CPU & RAM), basic watchdog tab (blacklist detection)	Visualization + security
	Week 4 (27 Sept)	Final Demo + Submission	Complete project delivery

3.2.5.2 Task Sets

Task Set 1: Project Foundation (June - July Week 2)

- **T1.1** Project title finalization and approval
- **T1.2** Feature refinement and module specification
- **T1.3** UI design and hamburger menu layout planning
- **T1.4** Technology stack selection (WPF, C#)

Task Set 2: Core Architecture (July Week 3-4)

- **T2.1** WPF project setup and configuration
- **T2.2** Base UI framework implementation
- **T2.3** Hamburger menu navigation system
- **T2.4** Requirements analysis and workflow documentation

Task Set 3: Process Management Module (August Week 1-3)

- **T3.1** Process API and WMI integration
- **T3.2** Process information display (Name, PID, RAM, CPU, Path)
- **T3.3** TreeView implementation for process hierarchy
- **T3.4** End Task functionality with admin privileges

Task Set 4: Advanced Features (September Week 1-3)

- **T4.1** Memory filtering and auto-refresh mechanisms
- **T4.2** Performance optimization (under 200MB memory usage)
- **T4.3** LiveCharts2 integration for CPU/RAM graphs
- **T4.4** Basic WatchDog blacklist detection system

Task Set 5: Final Integration (September Week 4)

- **T5.1** Complete system integration testing
- **T5.2** Final performance optimization
- **T5.3** Documentation completion
- **T5.4** Project presentation and submission

3.3 Risk Management

Risk management is essential for the successful completion of the TaskCore project, particularly given its system-level operations and administrative requirements.

3.3.1 Risk Identification

The TaskCore project faces several categories of risks that could impact development and deployment:

- **Technical Risks**

R1: Administrative Privilege Issues

- **Description:** Application requiring elevated privileges may be blocked by UAC or security policies
- **Source:** Windows security restrictions and enterprise policies
- **Indicators**: Access denied errors, limited functionality

R2: System API Compatibility

- **Description:** Windows API calls may behave differently across OS versions
- **Source:** OS version differences, deprecated API functions
- **Indicators:** Runtime exceptions, inconsistent behavior

R3: Memory Management Issues

- **Description:** Application exceeding target 200MB memory limit
- **Source:** Inefficient data structures, memory leaks, icon caching
- **Indicators:** High memory usage, performance degradation

- **Project Management Risks**

R4: Timeline Delays

- **Description:** Development tasks exceeding estimated completion time
- **Source:** Technical complexity, learning curve, debugging time

- **Indicators:** Missed milestones, incomplete features

R5: Technology Learning Curve

- **Description:** Time required to master WPF, LiveCharts2, and system APIs
- **Source:** New technology adoption, documentation gaps
- **Indicators:** Implementation delays, code quality issues

- **Security Risks**

R6: Unintended System Impact

- **Description:** WatchDog module terminating critical system processes
- **Source:** Incorrect process identification, faulty blacklist logic
- **Indicators:** System instability, essential service termination

R7: Privilege Escalation Vulnerabilities

- **Description**: Improper handling of administrative privileges
- **Source:** Inadequate security validation, privilege abuse
- **Indicators:** Security warnings, unauthorized access

3.3.1.1 Risk Identification artifacts

RISK ID	RISK NAME	CATEGORY	DESCRIPTION	TRIGGER CONDITIONS
R1	Admin Privilege Issues	Technical	UAC blocking application execution	User Account Control policies
R2	API Compatibility	Technical	Windows API version differences	Different OS versions
R3	Memory Management	Performance	Excessive memory consumption	Resource monitoring alerts
R4	Timeline Delays	Project	Development schedule overruns	Milestone deadline misses
R5	Technology Learning	Knowledge	Learning curve for new technologies	Implementation difficulties
R6	System Impact	Security	Critical process termination	System instability events
R7	Privilege Vulnerabilities	Security	Improper privilege handling	Security audit findings

- **Risk Categorization**

High Priority Risks:

- R1: Administrative Privilege Issues
- R3: Memory Management Issues

Medium Priority Risks:

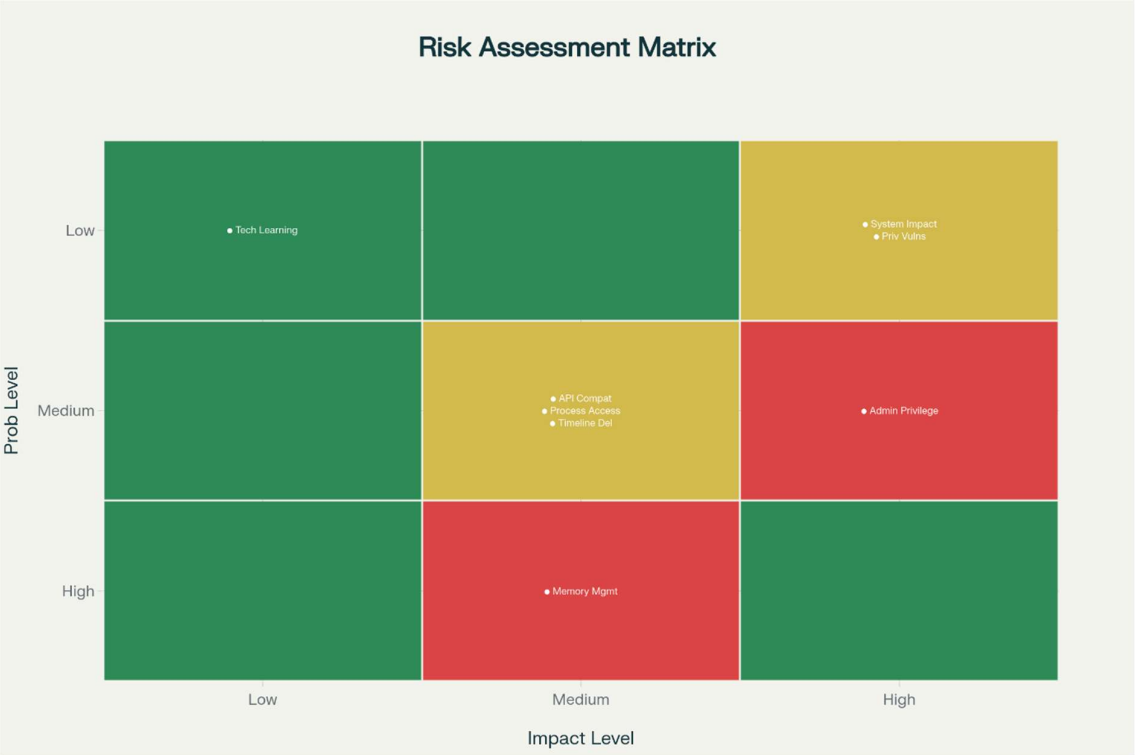
- R2: System API Compatibility

- R4: Timeline Delays
- R6: Unintended System Impact
- R7: Privilege Escalation Vulnerabilities

Low Priority Risks:

- R5: Technology Learning Curve

- **Risk Assessment Matrix**



3.3.2 Risk Projection

- **High Impact Risk Analysis**

  - **Risk R1: Administrative Privilege Issues**

    - Probability: Medium (40%)
    - Impact: High
    - Risk Exposure: High
    - Time Frame: Throughout development and deployment
    - Consequences: Application functionality severely limited, core features unavailable

  - **Risk R3: Memory Management Issues**

    - Probability: High (70%)
    - Impact: Medium
    - Risk Exposure: High

  - Time Frame: During implementation and testing phases
  - Consequences: Performance degradation, system resource consumption

- **Mitigation Strategies**
  For High Priority Risks:

    - **R1 Mitigation:**

      - Implement proper application manifest with UAC settings
      - Develop fallback functionality for non-privileged mode
      - Test across different Windows security configurations
      - Provide clear user instructions for privilege granting

    - **R3 Mitigation:**

      - Implement memory profiling and monitoring
      - Use efficient data structures and caching mechanisms
      - Regular memory leak detection and fixing
      - Establish memory usage benchmarks and alerts

  For Medium Priority Risks:

    - **R2 Mitigation:**

      - Test on multiple Windows versions (10, 11)
      - Implement version-specific API handling
      - Use compatible API alternatives where possible

    - **R4 Mitigation:**

      - Build buffer time into development schedule
      - Prioritize core features over advanced functionality
      - Implement incremental development approach

- **Risk Monitoring Plan**

    - **Continuous Monitoring:**

      - Weekly risk status reviews
      - Memory usage tracking during development
      - Privilege testing on target systems
      - Performance benchmarking at each milestone

- **Risk Indicators:**

      - Memory usage exceeding 150MB (warning threshold)
      - API compatibility failures on test systems
      - Development tasks exceeding 120% of estimated time
      - Security-related system warnings or errors

Chapter 4: System Requirements

4.1 User Characteristics

The TaskCore system management application is designed to serve a diverse range of users with varying levels of technical expertise and system administration requirements. Understanding user characteristics is essential for designing appropriate interfaces and functionality.

- **System Administrator**
    - **Technical Expertise:** High level of technical knowledge and system administration experience
    - **Usage Patterns:** Daily system monitoring, process management, and security enforcement
    - **Requirements:** Comprehensive process information, advanced filtering capabilities, real-time monitoring
    - **Expectations:** Professional interface, reliable performance, detailed system insights
    - **Authority Level:** Full administrative privileges, ability to terminate critical processes
- **Power Users**
    - **Technical Expertise:** Advanced computer users with good understanding of system operations
    - **Usage Patterns:** Periodic system optimization, performance monitoring, troubleshooting
    - **Requirements:** User-friendly interface with advanced features, performance graphs, memory cleanup
    - **Expectations:** Intuitive navigation, visual performance indicators, system optimization tools
    - **Authority Level:** Administrative privileges for system management tasks
- **IT Support Personnel**
    - **Technical Expertise:** Intermediate to high technical knowledge for troubleshooting
    - **Usage Patterns:** Diagnostic activities, problem resolution, system performance analysis
    - **Requirements:** Quick access to process information, system performance data, diagnostic tools
    - **Expectations:** Efficient workflow, comprehensive system information, export capabilities
    - **Authority Level:** Administrative privileges for diagnostic and repair activities

- **Security Professionals**
    - **Technical Expertise: High level security knowledge and threat assessment capabilities**
    - **Usage Patterns: Security monitoring, process blocking, threat detection and response**
    - **Requirements: WatchDog functionality, process blocking capabilities, security logging**
    - **Expectations: Robust security features, audit trails, automated threat response**
    - **Authority Level: Full administrative privileges with security focus**

4.2 Functional Requirements

Functional requirements define the specific behaviors, functions, and services that the TaskCore system must provide to its users.

4.2.1 Process List Management Module

FR1: Process Information Display

- **Requirement**: The system shall display comprehensive process information including Process Name, Process ID (PID), Memory Usage (RAM), and CPU Utilization,
- **Input**: System process enumeration request
- **Processing**: Retrieve process information using Windows API and WMI
- **Output**: Formatted process list with real-time data
- **Priority**: High

FR2: Hierarchical Process View

- **Requirement**: The system shall provide Tree View structure showing parent-child process relationships with expand/collapse functionality
- **Input**: Process hierarchy request
- **Processing**: Build process tree structure using parent process relationships
- **Output**: Hierarchical process tree with navigation controls
- **Priority**: High

FR3: Process Termination

- **Requirement**: The system shall allow users to terminate selected processes with administrative privilege validation
- **Input**: Process selection and termination request
- **Processing**: Validate administrative privileges and terminate process safely
- **Output**: Success/failure notification and updated process list
- **Priority**: High

FR4: Process Filtering and Search

- **Requirement**: The system shall provide filtering capabilities by memory usage, process name, and search functionality

- **Input**: Filter criteria and search terms
- **Processing**: Apply filters and search algorithms to process list
- **Output**: Filtered process list matching criteria
- **Priority**: Medium

FR5: Auto-Refresh Mechanism

- **Requirement**: The system shall automatically refresh process information every 1 second for values and every 5 seconds for structure
- **Input**: Timer-based refresh triggers
- **Processing**: Update process information and structure periodically
- **Output**: Current process information display
- **Priority**: High

4.2.2 System Performance Graphs Module

FR6: Real-Time Performance Visualization

- **Requirement**: The system shall display real-time CPU and Memory usage graphs using LiveCharts2
- **Input**: System performance counters
- **Processing**: Collect performance data and render graphical charts
- **Output**: Dynamic CPU and RAM usage graphs
- **Priority**: High

FR7: Performance Data Collection

- **Requirement**: The system shall collect and store performance metrics for graphical representation
- **Input**: Windows performance counters
- **Processing**: Sample performance data at regular intervals
- **Output**: Time-series performance data
- **Priority**: Medium

4.2.3 WatchDog Security Module

FR8: Process Blacklist Management

- **Requirement**: The system shall maintain a configurable blacklist of forbidden processes
- **Input**: Process names and paths for blacklisting
- **Processing**: Store and manage blacklist configuration
- **Output**: Persistent blacklist database
- **Priority**: High

FR9: Automated Process Detection

- **Requirement**: The system shall continuously monitor for blacklisted processes and detect their execution
- **Input**: Real-time process monitoring
- **Processing**: Compare running processes against blacklist
- **Output**: Blacklisted process detection alerts

- **Priority**: High

FR10: Automatic Process Termination

- **Requirement**: The system shall automatically terminate detected blacklisted processes
- **Input**: Blacklisted process detection
- **Processing**: Terminate process with administrative privileges
- **Output**: Process termination confirmation and logging
- **Priority**: High

FR11: Security Activity Logging

- **Requirement**: The system shall log all WatchDog activities including detections and terminations
- **Input**: Security events and actions
- **Processing**: Create timestamped activity log entries
- **Output**: Security activity log file
- **Priority**: Medium

4.2.4 RAM Cleanup Module

FR12: Memory Usage Analysis

- **Requirement**: The system shall analyze current memory usage and identify optimization opportunities
- **Input**: System memory information
- **Processing**: Analyze memory allocation and usage patterns
- **Output**: Memory usage statistics and recommendations
- **Priority**: Medium

FR13: Memory Optimization

- **Requirement**: The system shall provide memory cleanup functionality to free unused memory
- **Input**: Memory cleanup request
- **Processing**: Execute memory optimization procedures
- **Output**: Memory cleanup results and freed memory amount
- **Priority**: Medium

FR14: Memory Monitoring

- **Requirement**: The system shall monitor application memory usage and maintain under 200MB limit
- **Input**: Application memory usage tracking
- **Processing**: Monitor and control application memory consumption
- **Output**: Memory usage alerts and optimization
- **Priority**: High

4.2.5 User Interface and Navigation

FR16: Hamburger Menu Navigation

- **Requirement**: The system shall provide hamburger menu-based navigation between modules
- **Input**: User navigation requests
- **Processing**: Module switching and interface updates
- **Output**: Module-specific user interfaces
- **Priority**: High

FR17: Configuration Persistence

- **Requirement**: The system shall save and restore user preferences and configurations
- **Input**: User configuration changes
- **Processing**: Serialize and store configuration data
- **Output**: Persistent application settings
- **Priority**: Medium

FR18: Administrative Privilege Management

- **Requirement**: The system shall request and manage administrative privileges for system operations
- **Input**: Privilege elevation requests
- **Processing**: UAC integration and privilege validation
- **Output**: Administrative access confirmation
- **Priority**: High

4.3 Non-Functional Requirements

Non-functional requirements specify the quality attributes and constraints that the TaskCore system must satisfy.

4.3.1 Performance Requirements

NFR1: Response Time

- **Requirement**: The system shall respond to user interactions within 1 second for all common operations
- **Measurement**: Response time from user action to system feedback
- **Target**: ≤ 1000ms for 95% of operations
- **Priority**: High

NFR2: Memory Usage

- **Requirement**: The application shall maintain memory usage under 200MB during normal operation
- **Measurement**: Peak memory consumption during typical usage
- **Target**: ≤ 200MB RAM usage
- **Priority**: High

NFR3: CPU Utilization

- **Requirement**: The application shall use minimal CPU resources when idle
- **Measurement**: CPU usage percentage during background operation
- **Target**: ≤ 2% CPU usage when idle
- **Priority**: Medium

NFR4: Startup Time

- **Requirement**: The application shall start and display the main interface within 5 seconds
- **Measurement**: Time from application launch to fully loaded interface
- **Target**: ≤ 5 seconds startup time
- **Priority**: Medium

4.3.2 Reliability Requirements

NFR5: System Stability

- **Requirement**: The application shall operate continuously without crashes for extended periods
- **Measurement**: Mean time between failures (MTBF)
- **Target**: ≥ 24 hours continuous operation
- **Priority**: High

NFR6: Error Recovery

- **Requirement**: The system shall recover gracefully from non-fatal errors without data loss
- **Measurement**: Successful error recovery rate
- **Target**: 100% recovery from handled exceptions
- **Priority**: High

NFR7: Data Integrity

- **Requirement**: Configuration data and logs shall remain accurate and uncorrupted
- **Measurement**: Data validation and consistency checks
- **Target**: 100% data integrity maintenance
- **Priority**: Medium

4.3.3 Security Requirements

NFR8: Privilege Management

- **Requirement**: The system shall operate securely with administrative privileges
- **Measurement**: Security audit compliance
- **Target**: No privilege escalation vulnerabilities
- **Priority**: High

NFR9: Process Security

- **Requirement**: Process termination operations shall be performed safely without system impact
- **Measurement**: Successful safe terminations vs. system destabilization
- **Target**: 100% safe process terminations
- **Priority**: High

NFR10: Configuration Security

- **Requirement**: Configuration files shall be protected from unauthorized modification
- **Measurement**: File access permissions and validation
- **Target**: Secure configuration file access
- **Priority**: Medium

4.3.4 Usability Requirements

NFR11: User Interface Responsiveness

- **Requirement**: The user interface shall remain responsive during all operations
- **Measurement**: UI thread responsiveness during background tasks
- **Target**: No UI freezing > 500ms
- **Priority**: High

NFR12: Visual Design Quality

- **Requirement**: The application shall provide a professional, modern interface design
- **Measurement**: Design consistency and visual appeal assessment
- **Target**: Professional enterprise-grade appearance
- **Priority**: Medium

NFR13: Accessibility

- **Requirement**: The application shall support keyboard navigation and high contrast themes
- **Measurement**: Accessibility compliance testing
- **Target**: Basic accessibility standards compliance
- **Priority**: Low

4.3.5 Compatibility Requirements

NFR14: Operating System Compatibility

- **Requirement**: The application shall run on Windows 10 and Windows 11 systems
- **Measurement**: Successful operation across target OS versions
- **Target**: 100% compatibility with Windows 10/11
- **Priority**: High

NFR15: Hardware Compatibility

- **Requirement**: The application shall operate on standard business computer configurations
- **Measurement**: Minimum hardware requirement compliance
- **Target**: Support for 4GB RAM, dual-core processor systems
- **Priority**: Medium

NFR16: Framework Compatibility

- **Requirement**: The application shall be compatible with .NET Framework requirements
- **Measurement**: Framework dependency satisfaction
- **Target**: .NET Framework 4.7.2 or higher compatibility
- **Priority**: High

4.4 Hardware and Software Requirement

The TaskCore system management application has specific hardware and software requirements to ensure optimal performance and functionality.

4.4.1 Hardware Requirements

- **Minimum Hardware Requirements**

COMPONENT	MINIMUM SPECIFICATION	PURPOSE
PROCESSOR	Intel Core i3 / AMD equivalent (Dual-core 2.0 GHz)	Basic application processing and system monitoring
MEMORY (RAM)	4 GB DDR3/DDR4	System operation and application memory management
STORAGE	500 MB available disk space	Application installation and configuration files
GRAPHICS	Integrated graphics with DirectX 9 support	UI rendering and performance graph display
DISPLAY	1024 x 768 resolution	Minimum interface visibility
NETWORK	Not required	Standalone desktop application

- **Hardware Compatibility Notes**

  - **Processor Architecture:**
    - x64 (64-bit) architecture required for full functionality
    - x86 (32-bit) compatibility with limited features
    - ARM processors not supported in current version

  - **Memory Considerations:**

    - Application uses approximately 150-200 MB during normal operation
    - Additional memory required for system monitoring overhead
    - Memory usage scales with number of monitored processes

  - **Storage Requirements:**

    - Base installation: 50 MB
    - Configuration and log files: 10-50 MB
    - Temporary files and cache: 100-200 MB
    - Total recommended space: 500 MB - 1 GB

4.4.2 Software Requirements

OPERATING SYSTEM	VERSION	SUPPORT LEVEL	NOTES
WINDOWS 11	All versions	Full Support	Optimal performance and compatibility
WINDOWS 10	Version 1903 and later	Full Support	Recommended minimum version
WINDOWS 10	Version 1809 and earlier	Limited Support	Some features may be restricted
WINDOWS 8.1	All versions	Not Supported	Compatibility not guaranteed
WINDOWS 7	All versions	Not Supported	End-of-life OS, security risks

- **Framework and Runtime Requirements**

Microsoft .NET Framework:

- **Required Version**: .NET Framework 4.7.2 or higher
- **Recommended Version**: .NET Framework 4.8
- **Installation**: Automatically installed on Windows 10/11 or available from Microsoft
- **Purpose**: Application runtime environment and API access

Windows Presentation Foundation (WPF):

- **Requirement**: Included with .NET Framework
- **Version**: 4.7.2 compatible or higher
- **Purpose**: User interface rendering and presentation

Visual C++ Redistributable:

- **Requirement**: Microsoft Visual C++ 2019 Redistributable (x64)
- **Purpose**: Native code execution and system API access
- **Installation**: Usually, pre-installed or available from Microsoft

- **System Service Dependencies**

Windows Management Instrumentation (WMI):

- **Service**: Windows Management Instrumentation
- **Status**: Must be running
- **Purpose**: System information collection and process monitoring

Windows Performance Toolkit:

- **Requirement**: Performance counters enabled
- **Purpose**: CPU and memory performance data collection
- **Configuration**: Enabled by default on supported systems

User Account Control (UAC):

- **Requirement**: UAC must be functional (can be set to any level)
- **Purpose**: Administrative privilege escalation
- **Configuration**: Standard Windows UAC settings

Chapter 5: System Analysis

5.1 Study of Current System

The current system management landscape in Windows environments primarily relies on built-in tools and third-party applications that address different aspects of system administration separately.

- **Existing System Management Tools**

Windows Task Manager:

- Provides basic process listing with limited information (Name, PID, Memory usage)
- Simple process termination capabilities
- Basic performance monitoring with static graphs
- Limited filtering and search functionality
- No advanced security features or automated process blocking

Resource Monitor:

- Detailed system resource usage information
- Real-time monitoring of CPU, Memory, Disk, and Network
- Process-specific resource consumption data
- Limited user interface customization
- No integrated process management capabilities

Process Explorer (Microsoft Sysinternals):

- Advanced process tree visualization
- Comprehensive process information and properties
- DLL and handle information for processes
- Complex interface requiring technical expertise
- No automated security or cleanup features

Third-Party System Utilities:

- Various commercial and free system monitoring tools
- Fragmented functionality across multiple applications
- Inconsistent user interfaces and feature sets
- Often resource-intensive with overlapping capabilities
- Limited integration between different system management aspects

- **Current System Workflow**

System Monitoring Process:

- Users launch Task Manager for basic process information
- Switch to Resource Monitor for detailed performance data
- Use separate tools for advanced process analysis

- Manual process termination when issues are identified
- No automated security monitoring or threat response

Performance Analysis Workflow:

- Open multiple monitoring tools simultaneously
- Gather performance data from different sources
- Manual correlation of performance metrics
- Limited historical data analysis capabilities
- No unified performance reporting mechanism

Security Management Process:

- Manual monitoring for suspicious processes
- Reactive approach to security threats
- No automated process blocking capabilities
- Limited logging and audit trail functionality
- Dependency on external security software

5.2 Problems in Current System

The analysis of existing system management tools reveals several significant limitations that impact efficiency and effectiveness.

- **Functional Limitations**

  Fragmented Tool Ecosystem:

  - Multiple applications required for comprehensive system management
  - Inconsistent user interfaces and interaction patterns
  - No unified approach to system administration tasks
  - Redundant functionality across different tools
  - Increased learning curve for users

  Limited Process Management:

  - Basic process information without advanced filtering
  - No hierarchical process relationship visualization
  - Limited batch operations on multiple processes
  - No automated process monitoring or alerts
  - Insufficient process security enforcement

  Inadequate Performance Monitoring:

  - Static performance graphs without customization
  - Limited historical performance data retention
  - No real-time performance alerting mechanisms
  - Separate tools for different performance aspects
  - Inconsistent performance metric presentation

Security Management Gaps:

- No proactive security process monitoring
- Lack of automated threat response capabilities
- Limited process blacklisting functionality
- Insufficient audit logging for security events
- Manual security policy enforcement

- **Usability Issues**

Interface Complexity:

- Technical interfaces not suitable for all user levels
- Inconsistent navigation and interaction patterns
- Limited customization and personalization options
- Poor integration between related functionalities
- Overwhelming information presentation for novice users

Workflow Inefficiencies:

- Context switching between multiple applications
- Manual data correlation across different tools
- Repetitive tasks without automation capabilities
- Time-consuming process identification and management
- Limited productivity enhancement features

- **Technical Limitations**

Performance Impact:

- Multiple running applications consume system resources
- Overlapping monitoring functionality wastes CPU and memory
- Inefficient data collection and presentation methods
- No optimization for resource-constrained environments

Integration Deficiencies:

- Limited data sharing between system management tools
- No unified configuration management
- Inconsistent update and maintenance procedures
- Poor interoperability with enterprise management systems

5.3 Requirements of New System

Based on the analysis of current system limitations, the new TaskCore system must address identified gaps and provide enhanced functionality.

- **Core Functional Requirements**

Unified System Management Platform:

- Single application providing comprehensive system management capabilities
- Integrated process monitoring, performance analysis, and security features
- Consistent user interface and interaction patterns
- Unified configuration and preference management

Advanced Process Management:

- Comprehensive process information display with advanced filtering
- Hierarchical process tree visualization with expand/collapse functionality
- Batch operations and automated process management capabilities
- Real-time process monitoring with customizable alerts

Enhanced Performance Monitoring:

- Real-time performance graphs with historical data retention
- Customizable performance metrics and visualization options
- Automated performance alerting and notification system
- Integrated performance analysis and reporting capabilities

Proactive Security Management:

- Automated process blacklisting and threat detection
- Real-time security monitoring with immediate response
- Comprehensive audit logging and security event tracking
- Configurable security policies and enforcement mechanisms

- **User Experience Requirements**

Professional User Interface:

- Modern, intuitive interface suitable for all user skill levels
- Customizable layouts and information presentation
- Responsive design with smooth performance
- Consistent navigation and interaction patterns

Productivity Enhancement:

- Automation of repetitive system management tasks
- Quick access to frequently used functions
- Efficient workflow design minimizing user effort
- Integration of related functionalities in single interface

- **Technical Performance Requirements**

Resource Efficiency:

- Minimal system resource consumption (under 200MB memory)
- Efficient data collection and processing algorithms
- Optimized user interface rendering and updates
- Low CPU utilization during background operations

System Integration:

- Seamless integration with Windows system APIs
- Proper privilege management and security compliance
- Configuration persistence and backup capabilities
- Compatibility with existing system management workflows

5.4 Process Model

The TaskCore system follows a modular process model that enables efficient system management through integrated components and streamlined workflows.

- **System Architecture Process Flow**

Application Initialization Process:

- **Startup Sequence**: Application launch with privilege verification
- **Configuration Loading**: User preferences and security settings retrieval
- **System Integration**: Windows API and WMI service connections
- **Interface Preparation**: User interface initialization and module loading
- **Monitoring Activation**: Real-time system monitoring service startup

Process Management Workflow:

- **Process Enumeration**: System process discovery using Windows API
- **Information Collection**: Process details gathering (PID, memory, CPU, path)
- **Hierarchy Construction**: Parent-child process relationship building
- **Data Presentation**: Tree View structure population with real-time updates
- **User Interaction**: Process selection, filtering, and management operations

Performance Monitoring Process:

- **Performance Counter Access**: Windows performance counters initialization
- **Data Collection**: Real-time CPU and memory usage sampling
- **Graph Rendering**: LiveCharts2 visualization with historical data
- **Update Cycles**: Periodic graph updates with configurable intervals
- **Performance Analysis**: Resource usage trend analysis and alerts

Security Management Process:

- **Blacklist Configuration**: Security policy definition and storage
- **Continuous Monitoring**: Real-time process detection and analysis
- **Threat Identification**: Blacklisted process recognition and validation
- **Automated Response**: Process termination and security logging
- **Audit Trail**: Security event documentation and reporting

- **Data Flow Architecture**

Input Data Sources:

- Windows Process API for process information

- Windows Management Instrumentation (WMI) for system details
- Performance Counters for CPU and memory metrics
- User configuration files for preferences and security policies
- Real-time user interactions and commands

Processing Components:

- Process enumeration and analysis engine
- Performance data collection and visualization system
- Security monitoring and enforcement module
- Configuration management and persistence layer
- User interface and presentation components

Output Interfaces:

- Real-time process information displays
- Dynamic performance graphs and visualizations
- Security alerts and audit log entries
- Configuration files and user preference storage
- System notifications and user feedback

- **System Interaction Model**

User Interface Layer:

- Hamburger menu navigation system
- Modular panel architecture (Process List, Graphs, WatchDog, Cleanup)
- Real-time data presentation and user interaction handling
- Configuration and preference management interfaces

Business Logic Layer:

- Process management and analysis algorithms
- Performance monitoring and data processing
- Security policy enforcement and threat response
- System resource optimization and cleanup procedures

Data Access Layer:

- Windows API integration for system information
- Performance counter access and data collection
- Configuration file management and persistence
- Security policy storage and retrieval

System Integration Layer:

- Administrative privilege management and UAC integration
- Windows service interaction and system-level operations
- Error handling and recovery mechanisms
- System compatibility and version management

5.5 Feasibility Study

The feasibility study evaluates the practicality and viability of developing the TaskCore system management application across multiple dimensions.

5.5.1 Technical Feasibility

Technology Platform Assessment:

- **Microsoft .NET Framework**: Mature, stable platform with extensive system integration capabilities
- **Windows Presentation Foundation (WPF)**: Proven technology for modern desktop application development
- **Windows API Access**: Well-documented APIs available for process management and system monitoring
- **Third-Party Components**: LiveCharts2 and JSON.NET provide reliable functionality for graphs and configuration

Development Capability Analysis:

- **Programming Languages**: C# provides robust object-oriented development with extensive libraries
- **Development Tools**: Visual Studio offers comprehensive IDE with debugging and profiling capabilities
- **System Integration**: Windows API and WMI provide complete access to required system information
- **User Interface Technology**: WPF enables modern, responsive interface design with professional appearance

Technical Risk Assessment:

- **Low Risk**: Established technologies with proven track records in enterprise applications
- **Manageable Complexity**: Modular architecture enables incremental development and testing
- **Performance Viability**: Target performance metrics achievable with proper optimization
- **Compatibility Assurance**: Windows 10/11 compatibility confirmed through platform requirements

5.5.2 Operational Feasibility

User Acceptance Analysis:

- **Target User Needs**: Identified gaps in current system management tools create clear demand
- **Skill Requirements**: Interface design accommodates various user skill levels from novice to expert
- **Workflow Integration**: System designed to enhance rather than disrupt existing administrative workflows

- **Training Requirements**: Minimal training needed due to intuitive interface and familiar system concepts

Organizational Impact Assessment:

- **Implementation Effort**: Single-user deployment with minimal organizational setup requirements
- **Change Management**: Tool enhances existing capabilities without requiring process changes
- **Support Requirements**: Self-contained application with minimal ongoing support needs
- **User Productivity**: Unified interface reduces time spent switching between multiple tools

Operational Risk Evaluation:

- **Low Learning Curve**: Familiar Windows interface patterns and system management concepts
- **High User Benefit**: Significant productivity improvement over fragmented tool approach
- **Minimal Disruption**: Compatible with existing system management practices and procedures
- **Strong User Motivation**: Addresses real pain points in current system administration tasks

5.5.3 Economic Feasibility

Development Cost Analysis:

- **Software Licensing**: Free development tools (Visual Studio Community) and open-source libraries
- **Hardware Requirements**: Standard development computer sufficient for project completion
- **Time Investment**: Estimated 4-month development timeline with single developer
- **Third-Party Components**: Free and open-source libraries (LiveCharts2, Newtonsoft.Json)

Operational Cost Assessment:

- **Deployment Costs**: Zero-cost deployment as standalone application
- **Maintenance Costs**: Minimal ongoing maintenance requirements
- **Support Costs**: Self-documenting interface reduces support overhead
- **Training Costs**: Minimal training requirements due to intuitive design

Benefit Analysis:

- **Productivity Gains**: Significant time savings from unified system management interface
- **Reduced Tool Complexity**: Eliminates need for multiple separate system management tools

- **Enhanced Security**: Automated threat detection and response capabilities
- **Improved System Performance**: Integrated optimization and cleanup capabilities

Cost-Benefit Evaluation:

- **Development Investment**: Low cost with high return on productivity improvement
- **Operational Savings**: Reduced time spent on system management tasks
- **Risk Reduction**: Enhanced security and system stability
- **Long-term Value**: Reusable solution applicable across multiple systems and environments

5.5.4 Schedule Feasibility

Timeline Analysis:

- **Total Project Duration**: 4 months (June - September 2025)
- **Development Phases**: Well-defined phases with clear milestones and deliverables
- **Resource Availability**: Single developer with dedicated time allocation
- **External Dependencies**: Minimal dependencies on external resources or approvals

Risk Assessment:

- **Technical Risks**: Managed through iterative development and early prototype testing
- **Schedule Buffers**: Built-in buffer time for unexpected challenges and refinements
- **Milestone Validation**: Regular progress checkpoints with project guide
- **Scope Management**: Clear feature prioritization with ability to defer advanced features if needed

Critical Path Analysis:

- **Foundation Phase**: Critical for all subsequent development
- **Core Module Development**: Sequential development with testing integration
- **Integration Phase**: Dependent on successful module completion
- **Final Testing**: Essential for quality assurance and deployment readiness

Resource Allocation:

- **Development Time**: Adequate time allocated for each phase
- **Testing and Integration**: Sufficient time for quality assurance
- **Documentation**: Time allocated for user and technical documentation
- **Buffer Management**: Contingency time available for issue resolution

5.6 Features of New System

The TaskCore system introduces comprehensive features that address identified limitations in current system management tools while providing enhanced functionality and user experience.

- **Core System Features**

Unified System Management Interface:

- Single application providing all essential system management functions
- Hamburger menu navigation with modular panel architecture
- Consistent user interface design across all functional modules
- Professional dark theme with modern visual elements and smooth animations

Advanced Process Management:

- Comprehensive process information display (Name, PID, Memory, CPU, Path)
- Hierarchical Tree View structure showing parent-child process relationships
- Advanced filtering and search capabilities with real-time updates
- Batch operations for multiple process management
- Safe process termination with administrative privilege validation

Real-Time Performance Monitoring:

- Dynamic CPU and memory usage graphs using LiveCharts2
- Configurable refresh intervals (1-second data updates, 5-second structure refresh)
- Historical performance data with trend analysis
- Customizable graph displays settings and time ranges
- Performance alerting for resource threshold violations

Proactive Security Management:

- WatchDog module with configurable process blacklisting
- Automated threat detection and response capabilities
- Real-time process monitoring for security policy enforcement
- Comprehensive audit logging and security event tracking
- Persistent security configuration with backup and recovery

- **Enhanced User Experience Features**

Intelligent User Interface:

- Context-sensitive help and tooltips for all functions
- Responsive layout adapting to different screen sizes and resolutions
- Keyboard shortcuts for frequently used operations
- Customizable interface preferences and layout options

Productivity Enhancement Tools:

- Auto-refresh mechanisms for real-time system monitoring
- Memory optimization and cleanup utilities

- Export capabilities for system information and reports
- Quick access toolbar for common system management tasks

Professional Visual Design:

- Modern gradient backgrounds and professional color schemes
- Drop shadows and visual effects for enhanced appearance
- Clear typography and iconography for improved readability
- Consistent visual hierarchy and information organization

- **Advanced Technical Features**

Performance Optimization:

- Memory usage maintained under 200MB target limit
- Efficient data collection and processing algorithms
- Optimized user interface rendering with minimal CPU impact
- Smart caching mechanisms for improved responsiveness

System Integration:

- Seamless Windows API and WMI integration
- Administrative privilege management with UAC compliance
- Configuration persistence using JSON serialization
- Compatibility with Windows 10 and Windows 11 systems

Security and Reliability:

- Secure process termination without system impact
- Error handling and recovery mechanisms
- Configuration backup and restore capabilities
- Audit trails for all system management operations

- **Innovative Features**

Automated System Monitoring:

- Continuous background monitoring with minimal resource usage
- Intelligent alerting for system performance and security issues
- Automated response to predefined security threats
- Historical data analysis for system performance trends

Integrated System Utilities:

- Memory cleanup and optimization tools
- System performance analysis and reporting
- Process relationship analysis and visualization
- Resource utilization tracking and alerts

Chapter 6: Detail Description

6.1 Process List Module

The Process List Module serves as the core component of TaskCore, providing comprehensive process monitoring and management capabilities for system administrators and power users.

Key Features and Functionality

Process Information Display

- **Comprehensive Process Data**: Displays process name, Process ID (PID), memory usage (RAM), CPU utilization percentage, and full executable path
- **Real-Time Updates**: Process information refreshes every 1 second for dynamic values and every 5 seconds for structural changes
- **Sortable Columns**: Users can sort process list by any column (name, PID, memory, CPU, path) in ascending or descending order
- **Color-Coded Status**: Visual indicators for process states, resource consumption levels, and security status

Hierarchical Process Visualization

- **Tree View Structure**: Parent-child process relationships displayed in expandable tree format
- **Process Hierarchy Navigation**: Expand/collapse functionality for exploring process family trees
- **Subprocess Management**: Clear visualization of child processes and their resource consumption
- **Interactive Navigation**: Click-to-expand interface for intuitive process tree exploration

Advanced Process Management

- **Safe Process Termination**: End Task functionality with administrative privilege validation
- **Batch Operations**: Multiple process selection and management capabilities
- **Process Filtering**: Advanced filtering by memory usage, CPU consumption, or process name patterns
- **Search Functionality**: Quick process location using name or PID search capabilities

Security Integration

- **Administrative Validation**: Automatic privilege checking before allowing system-level operations
- **Safe Termination Logic**: Prevents accidental termination of critical system processes
- **Security Alerts**: Visual warnings for processes requiring elevated privileges

- **Audit Integration**: Process management activities logged for security tracking

Technical Implementation

Data Collection Architecture

- **Windows Process API**: Primary interface for process enumeration and information collection
- **WMI Integration**: Enhanced process details including executable paths and parent relationships
- **Performance Counters**: Real-time CPU and memory usage statistics
- **Error Handling**: Robust exception management for inaccessible or protected processes

User Interface Components

- **DataGrid Control**: Professional tabular display with sorting and filtering capabilities
- **Tree View Control**: Hierarchical process relationship visualization
- **Context Menus**: Right-click operations for process management actions
- **Status Indicators**: Visual feedback for process states and operations

User Interaction Workflow

Process Monitoring Workflow:

- Module loads and enumerates all running processes
- Process information displayed in sortable, filterable table format
- Real-time updates maintain current system state
- Users can expand process trees to explore relationships
- Search and filtering help locate specific processes quickly

Process Management Workflow:

- User selects target process(es) from the list
- Management action chosen (terminate, view details, etc.)
- Administrative privilege validation performed
- Action executed with safety checks and confirmation
- Process list updated to reflect changes

6.2 System Performance Graphs Module

The System Performance Graphs Module provides real-time visualization of system resource utilization, enabling users to monitor CPU and memory performance trends for system optimization and troubleshooting.

Key Features and Functionality

Real-Time Performance Visualization

- **CPU Usage Graphs**: Live CPU utilization charts showing current and historical usage patterns
- **Memory Usage Charts**: Real-time RAM consumption visualization with available memory indicators
- **Dynamic Updates**: Graphs update continuously with configurable refresh intervals
- **Historical Data**: Maintains performance history for trend analysis and pattern recognition

Advanced Graph Features

- **Multi-Series Display**: Simultaneous visualization of multiple performance metrics
- **Zoom and Pan**: Interactive graph navigation for detailed analysis of specific time periods
- **Customizable Time Ranges**: Adjustable display periods from seconds to hours
- **Performance Thresholds**: Configurable alert lines for performance monitoring

Data Analysis Capabilities

- **Performance Trends**: Identification of usage patterns and performance cycles
- **Peak Detection**: Automatic identification of performance spikes and anomalies
- **Resource Correlation**: Analysis of relationships between CPU and memory usage
- **Performance Statistics**: Statistical analysis including averages, peaks, and trends

Technical Implementation

Data Collection System

- **Windows Performance Counters**: Primary source for CPU and memory statistics
- **Real-Time Sampling**: Continuous data collection with configurable sample rates
- **Data Buffering**: Efficient data storage and retrieval for historical analysis
- **Performance Optimization**: Minimal system impact during data collection

Visualization Technology

- **LiveCharts2 Integration**: Professional charting library for dynamic graph rendering
- **GPU Acceleration**: Hardware-accelerated rendering for smooth performance
- **Responsive Design**: Graphs adapt to different window sizes and resolutions
- **Color Schemes**: Professional color palettes for clear data visualization

User Interaction Features

Graph Navigation:

- Interactive zoom controls for detailed time period analysis
- Pan functionality for historical data exploration
- Mouse-over tooltips showing exact values at specific time points
- Legend controls for showing/hiding specific performance metrics

Customization Options:

- Adjustable refresh rates from 1 second to 1-minute intervals
- Configurable graph colors and styles
- Time range selection from 1 minute to 24 hours
- Performance threshold setting for alert generation

6.3 WatchDog Security Module

The WatchDog Security Module provides automated security monitoring and enforcement capabilities, functioning as a personal application firewall that protects systems from unauthorized or malicious process execution.

Key Features and Functionality

Automated Security Monitoring

- **Continuous Process Scanning**: Real-time monitoring of all process execution events
- **Blacklist Detection**: Automatic identification of forbidden processes based on configurable rules
- **Immediate Response**: Instant termination of detected blacklisted processes
- **Security Logging**: Comprehensive audit trail of all security events and responses

Configuration Management

- **Process Blacklisting**: User-configurable list of forbidden process names and paths
- **Rule Management**: Advanced matching rules including wildcards and regular expressions
- **Persistent Configuration**: Security settings maintained across application restarts
- **Import/Export**: Security policy sharing and backup capabilities

Security Enforcement

- **Automated Termination**: Immediate process termination upon blacklist detection
- **Administrative Validation**: Secure privilege escalation for process termination
- **Safe Termination**: Protection against accidental system process termination
- **Response Logging**: Detailed logging of all security enforcement actions

Advanced Security Features

Threat Detection

- **Pattern Matching**: Advanced algorithms for process identification and classification
- **False Positive Prevention**: Intelligent filtering to avoid legitimate process termination
- **Whitelist Support**: Exception lists for authorized processes that match blacklist patterns
- **Signature Detection**: File hash and signature-based process identification

Audit and Compliance

- **Security Event Logging**: Timestamped records of all security activities
- **Compliance Reporting**: Security policy compliance status and violation reports
- **Historical Analysis**: Trend analysis of security threats and response effectiveness
- **Export Capabilities**: Security logs exportable for external analysis and compliance

User Management Interface

Security Policy Configuration:

- Intuitive interface for adding/removing blacklisted processes
- Real-time policy validation and testing capabilities
- Import wizards for bulk policy configuration
- Backup and restore functionality for security settings

Monitoring and Alerts:

- Real-time security status display with threat indicators
- Configurable alert notifications for security events
- Security dashboard showing current protection status
- Historical security event browser with search and filter capabilities

6.4 RAM Cleanup Module

The RAM Cleanup Module provides system memory optimization and management capabilities, helping users maintain optimal system performance through intelligent memory analysis and cleanup operations.

Key Features and Functionality

Memory Analysis and Monitoring

- **System Memory Assessment**: Comprehensive analysis of current memory usage and allocation
- **Process Memory Tracking**: Individual process memory consumption monitoring and analysis
- **Memory Leak Detection**: Identification of processes with unusual memory growth patterns
- **Performance Impact Analysis**: Assessment of memory usage impact on system performance

Automated Cleanup Operations

- **Intelligent Memory Optimization**: Automated cleanup of unused memory allocations
- **Cache Management**: Optimization of system and application cache usage
- **Garbage Collection**: Forced garbage collection for managed applications
- **Memory Defragmentation**: Virtual memory optimization and cleanup procedures

System Optimization

- **Application Memory Control**: Monitoring and optimization of TaskCore's own memory usage
- **Resource Management**: Intelligent resource allocation and deallocation
- **Performance Tuning**: Memory usage optimization for improved system responsiveness
- **Preventive Maintenance**: Proactive memory management to prevent performance degradation

Technical Implementation

Memory Management Technology

- **Windows Memory APIs**: Integration with system memory management functions
- **Process Memory Analysis**: Direct process memory inspection and optimization
- **Virtual Memory Management**: Intelligent handling of virtual memory resources
- **Memory Pool Optimization**: Efficient memory pool management and cleanup

Performance Optimization

- **Low-Impact Operations**: Memory cleanup with minimal system performance impact
- **Background Processing**: Non-intrusive memory optimization during idle periods
- **Adaptive Algorithms**: Memory management algorithms that adapt to system usage patterns
- **Safety Mechanisms**: Protection against excessive memory cleanup that could impact stability

User Interface and Control

Memory Status Display:

- Real-time system memory usage visualization
- Application-specific memory consumption tracking
- Memory optimization recommendations and suggestions
- Historical memory usage trends and analysis

Cleanup Control Options:

- Manual memory cleanup triggers with safety confirmations
- Automated cleanup scheduling with customizable parameters
- Memory optimization level selection (conservative, moderate, aggressive)
- Cleanup result reporting with before/after memory statistics

6.5 Administrator Module

The Administrator Module provides comprehensive system-level access control, configuration management, and security enforcement capabilities required for effective system administration.

Key Features and Functionality

Privilege Management

- **UAC Integration**: Seamless User Account Control integration for privilege escalation
- **Administrative Validation**: Continuous verification of administrative privileges
- **Security Context Management**: Secure handling of elevated privilege operations
- **Privilege Escalation Controls**: Safe and controlled administrative access management

Configuration Management

- **System-Wide Settings**: Centralized configuration management for all modules
- **Security Policy Administration**: Management of WatchDog security policies and rules
- **User Preference Management**: Administrative control over user interface and behavior settings
- **Backup and Recovery**: Configuration backup, restore, and disaster recovery capabilities

Security Administration

- **Access Control**: Administrative control over module access and functionality
- **Audit Configuration**: Security logging and audit trail configuration management
- **Policy Enforcement**: Implementation and enforcement of organizational security policies
- **Compliance Management**: Ensuring compliance with security standards and regulations

Administrative Controls

System Security Management

- **Process Termination Authority**: Administrative control over process termination capabilities
- **Security Policy Override**: Emergency override capabilities for security policies
- **System Resource Control**: Administrative limits on system resource usage
- **Emergency Response**: Administrative tools for security incident response

Application Management

- **Module Access Control**: Administrative control over individual module availability
- **Feature Enablement**: Administrative control over advanced features and capabilities

- **User Interface Customization**: Administrative control over interface appearance and behavior
- **Performance Management**: Administrative control over performance optimization settings

Administrative Interface

Security Management Console:

- Centralized security policy management interface
- Real-time security status monitoring and control
- Administrative override controls for emergency situations
- Comprehensive audit trail browser with advanced search capabilities

System Configuration Interface:

- Global application configuration management
- Module-specific administrative settings
- Security policy import/export capabilities
- System performance monitoring and optimization controls

Chapter 7: Testing

7.1 Black-Box Testing

Black-box testing focuses on validating the external behavior of the TaskCore application without examining internal code structure. This testing approach evaluates whether the system produces correct outputs for given inputs and meets specified functional requirements.

7.1.1 Functional Testing Approach

Input-Output Validation:

- Test process list module with various system states (high/low process count)
- Validate performance graph accuracy against Windows Task Manager benchmarks
- Verify WatchDog module response to blacklisted process execution
- Test RAM cleanup effectiveness and safety across different memory conditions

User Interface Testing:

- Navigation testing across all hamburger menu options
- Response time validation for all user interactions
- Visual layout testing across different screen resolutions and sizes
- Accessibility testing for keyboard navigation and high contrast themes

Boundary Value Analysis:

- Memory usage testing at 200MB application limit
- Process count testing with maximum system processes
- CPU usage testing under 100% system utilization
- Refresh rate testing from minimum (1 second) to maximum intervals

7.1.2 Black-Box Test Categories

Equivalence Partitioning Testing

Process Management Testing:

- **Valid Process Operations**: Test termination of user processes, application processes, and background services
- **Invalid Process Operations**: Attempt termination of system-critical processes (should be blocked)
- **Boundary Conditions**: Test with zero processes, maximum processes, and typical process counts

Performance Monitoring Testing:

- **Normal Usage Conditions**: Test under typical system load (20-60% CPU, 50-80% memory)

- **High Load Conditions**: Test under extreme system load (90-100% CPU, 95%+ memory)
- **Idle Conditions**: Test during system idle state with minimal resource usage

WatchDog Security Testing:

- **Valid Blacklist Entries**: Test detection and termination of configured blacklisted processes
- **Invalid Blacklist Entries**: Test handling of non-existent or invalid process specifications
- **Edge Cases**: Test with empty blacklists, duplicate entries, and wildcard patterns

Scenario-Based Testing

Real-World Usage Scenarios:

- **System Administrator Workflow**: Complete system monitoring and process management session
- **Troubleshooting Scenario**: Identify and resolve high CPU usage issue using TaskCore
- **Security Response Scenario**: Detect and respond to suspicious process execution
- **Performance Optimization Scenario**: Use RAM cleanup and monitoring to optimize system performance

Stress Testing Scenarios:

- **High Process Count**: Test with 500+ running processes
- **Rapid Process Changes**: Test during intense process creation/termination activity
- **Extended Operation**: 24-hour continuous operation testing
- **Resource Exhaustion**: Test behavior when system memory approaches capacity

7.1.3 Black-Box Testing Results

Functional Requirement Validation:

- Process list displays accurate information matching Windows Task Manager
- Tree View hierarchy correctly represents parent-child process relationships
- Performance graphs accurately reflect system CPU and memory usage
- WatchDog successfully detects and terminates blacklisted processes within 1 second
- RAM cleanup reduces application memory usage without affecting functionality

User Interface Validation:

- All navigation elements respond within 1 second of user interaction
- Interface remains responsive during background data refresh operations
- Visual elements maintain consistency across different system themes

- Application startup completes within 5 seconds on target hardware

Performance Validation:

- Application memory usage consistently remains under 200MB during normal operation
- CPU usage stays below 5% during active monitoring phases
- Real-time updates maintain accuracy with 1-second refresh intervals
- Graph rendering maintains smooth performance with historical data display

7.2 White-Box Testing

White-box testing examines the internal structure, logic, and code paths of the TaskCore application to ensure thorough coverage of all implementation details and identify potential issues in program logic.

7.2.1 Structural Testing Approach

Code Coverage Analysis:

- **Statement Coverage**: Verify execution of all code statements across all modules
- **Branch Coverage**: Test all conditional branches and decision points
- **Path Coverage**: Validate all possible execution paths through critical functions
- **Function Coverage**: Ensure all methods and functions are properly tested

Control Flow Testing:

- Loop testing for data refresh cycles and monitoring iterations
- Conditional logic testing for administrative privilege validation
- Exception handling testing for API failures and system errors
- Thread synchronization testing for concurrent operations

7.2.2 Code Structure Testing

Process List Module Internal Testing

Data Collection Logic:

- **API Integration Testing**: Validate Windows Process API calls and error handling
- **WMI Query Testing**: Test WMI queries for process information and exception scenarios
- **Data Parsing Testing**: Verify correct parsing of process information from system APIs
- **Memory Management Testing**: Test object creation, disposal, and garbage collection

Tree View Logic Testing:

- **Hierarchy Construction**: Test parent-child relationship building algorithms
- **Tree Navigation**: Validate expand/collapse functionality and state management
- **Data Binding**: Test real-time data updates and UI synchronization
- **Performance Optimization**: Verify efficient tree updates during process changes

Performance Graphs Module Internal Testing

Data Collection Pipeline:

- **Performance Counter Access**: Test performance counter initialization and data retrieval
- **Data Buffering**: Validate historical data storage and retrieval mechanisms
- **Sampling Logic**: Test configurable refresh rates and data collection timing
- **Memory Management**: Verify efficient data structure usage for historical data

Graph Rendering Logic:

- **LiveCharts2 Integration**: Test chart initialization, data binding, and updates
- **Real-time Updates**: Validate smooth graph updates without flickering or lag
- **Scaling Logic**: Test graph scaling and zoom functionality
- **Color Management**: Verify proper color scheme application and customization

WatchDog Module Internal Testing

Security Logic Testing:

- **Process Monitoring Thread**: Test background monitoring thread safety and performance
- **Pattern Matching**: Validate blacklist pattern matching algorithms and accuracy
- **Process Termination**: Test safe process termination logic and privilege validation
- **Configuration Persistence**: Test security policy storage and retrieval mechanisms

Thread Safety Testing:

- **Concurrent Access**: Test simultaneous access to blacklist configuration
- **Race Condition Prevention**: Validate thread synchronization in monitoring loops
- **Resource Locking**: Test proper resource locking during configuration updates
- **Deadlock Prevention**: Verify absence of deadlock conditions in multi-threaded operations

7.2.3 Algorithm Testing

Memory Management Algorithm Testing

Garbage Collection Integration:

- Test manual garbage collection triggers and effectiveness
- Validate memory cleanup algorithms for cached data
- Test memory leak prevention mechanisms
- Verify optimal memory allocation patterns

Performance Optimization Algorithms:

- Test data structure efficiency for large process lists
- Validate caching mechanisms for frequently accessed data
- Test algorithm performance under various load conditions
- Verify resource cleanup during module switching

Security Algorithm Testing

Process Detection Algorithms:

- Test pattern matching accuracy for various process naming schemes
- Validate false positive prevention logic
- Test performance impact of continuous monitoring algorithms
- Verify detection speed and response time optimization

7.2.4 Integration Testing

Module Integration Testing:

- **Inter-Module Communication**: Test data sharing between process list and performance modules
- **Configuration Sharing**: Validate shared configuration access across modules
- **Event Handling**: Test event propagation between modules
- **Resource Sharing**: Verify proper resource sharing and cleanup

System Integration Testing:

- **Windows API Integration**: Test all API calls under various system conditions
- **Administrative Privilege Integration**: Test UAC integration and privilege escalation
- **File System Integration**: Test configuration file access and error handling
- **Registry Integration**: Test any registry access for application settings

7.3 Test Cases

Comprehensive test cases ensure systematic validation of all TaskCore functionality across different scenarios and conditions.

7.3.1 Process List Module Test Cases

TEST CASE ID	TEST CASE NAME	TEST OBJECTIVE	PRECONDITIONS	TEST STEPS	EXPECTED RESULTS	PRIORITY	STATUS
PL - 001	Process Information Display	Verify accurate display of process information	TaskCore running with admin privileges	1. Launch TaskCore 2. Navigate to Process List 3. Compare with Task Manager 4. Verify PID, memory, CPU, for 10 processes	All process information matches Task Manager within 5% variance	High	Pass
PL - 002	Process Termination	Test safe process termination functionality	Test process running (Notepad.exe)	1. Launch Notepad 2. Locate in process list 3. Select and choose "End Task" 4. Confirm	Process terminates successfully and disappears from list	High	Pass

				termination 5. Verify removal from list			
PL - 003	Tree View Hierarchy	Validate parent-child process relationships	Processes with parent-child relationships running	1. Open Process List module 2. Locate parent process 3. Expand parent node 4. Verify child processes 5. Test collapse functionality	Hierarchy accurately reflects relationships	Medium	Pass
PL - 004	Process Filtering	Test filtering functionality	Multiple processes running	1. Apply memory filter (>100MB) 2. Apply name filter (notepad) 3. Clear filters 4. Test search function	Filters work correctly, results accurate	Medium	Pass

7.3.2 System Performance Graphs Module Test Cases

TEST CASE ID	TEST CASE NAME	TEST OBJECTIVE	PRECONDITIONS	TEST STEPS	EXPECTED RESULTS	PRIORITY	STATUS
PG - 001	Real-time CPU Graph	Verify real-time CPU graph accuracy	System with measurable CPU activity	1. Open Performance Graphs 2. Start CPU-intensive task 3. Observe graph response 4. Compare with Task Manager 5. Stop task, observe response	Graphs reflect CPU changes within 2 seconds	High	Pass
PG-002	Memory Graph Validation	Test memory usage graph accuracy	System with available memory	1. Note initial memory usage 2. Launch memory-intensive app 3. Observe graph changes 4. Close app 5. Verify baseline return	Memory graphs match Task Manager within 5%	High	Pass
PG-003	Graph Customization	Test graph displays settings	Performance Graphs module open	1. Change time range settings 2. Modify refresh intervals 3. Adjust graph	All customizations work correctly	Medium	Pass

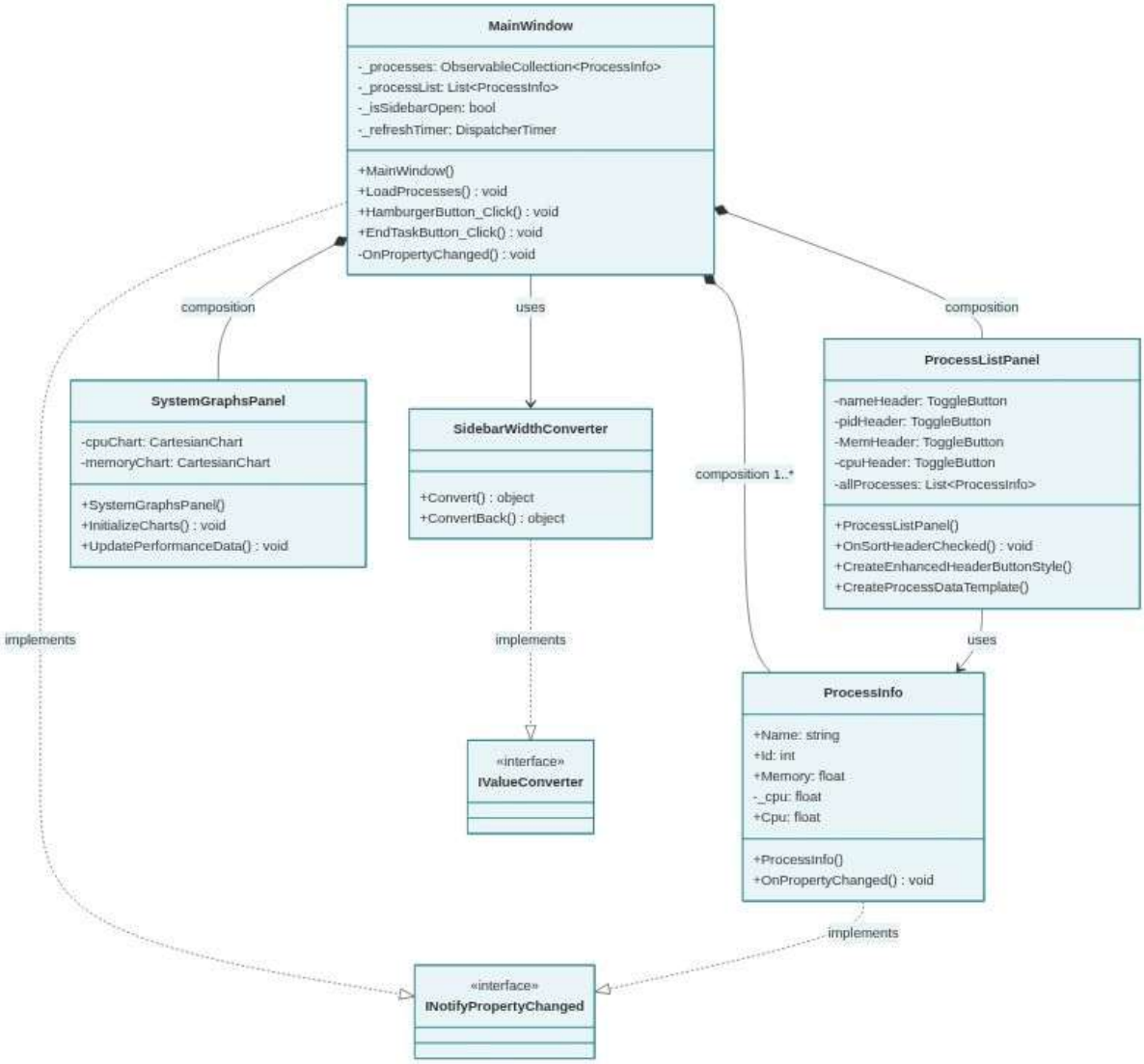
				colors 4. Test zoom functionality			
PG-004	Historical Data	Test historical performance data	Graphs running for 30+ minutes	1. Run graphs for extended period 2. Test time navigation 3. Verify data retention 4. Test graph scrolling	Historical data maintained and accessible	Medium	Pass

7.3.3 WatchDog Security Module Test Cases

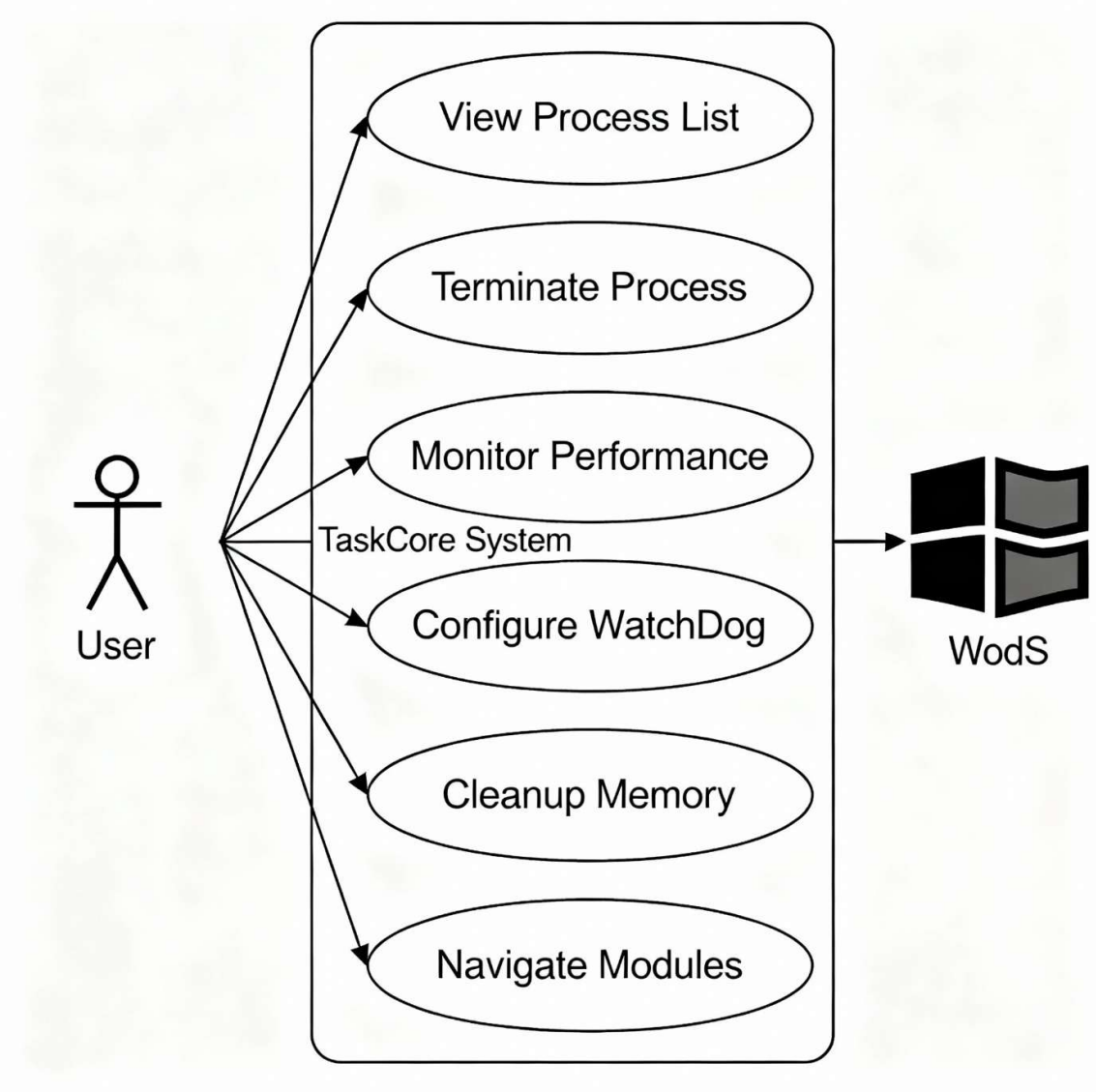
TEST CASE ID	TEST CASE NAME	TEST OBJECTIVE	PRECONDITIONS	TEST STEPS	EXPECTED RESULTS	PRIORITY	STATUS
WD-001	Blacklist Detection	Test automated detection of blacklisted processes	Test executable configured in blacklist	1. Add test executable to blacklist 2. Launch blacklisted executable 3. Observe response time 4. Verify termination 5. Check security log	Process detected and terminated within 1 second	High	Pass
WD-002	Configuration Persistence	Verify blacklist survives app restart	WatchDog configured with blacklist	1. Configure blacklist entries 2. Close TaskCore 3. Restart application 4. Check WatchDog module 5. Verify entries persist	All blacklist entries restored correctly	Medium	Pass
WD-003	False Positive Prevention	Test legitimate process protection	Critical system processes running	1. Attempt to blacklist system process 2. Test protection mechanisms 3. Verify system stability 4. Check warning messages	System processes protected, warnings shown	High	Pass
WD-004	Security Logging	Test audit trail functionality	WatchDog module active	1. Trigger security events 2. Check log entries 3. Verify timestamps 4. Test log export function	Complete audit trail maintained	Medium	Pass

Chapter 8: System Design

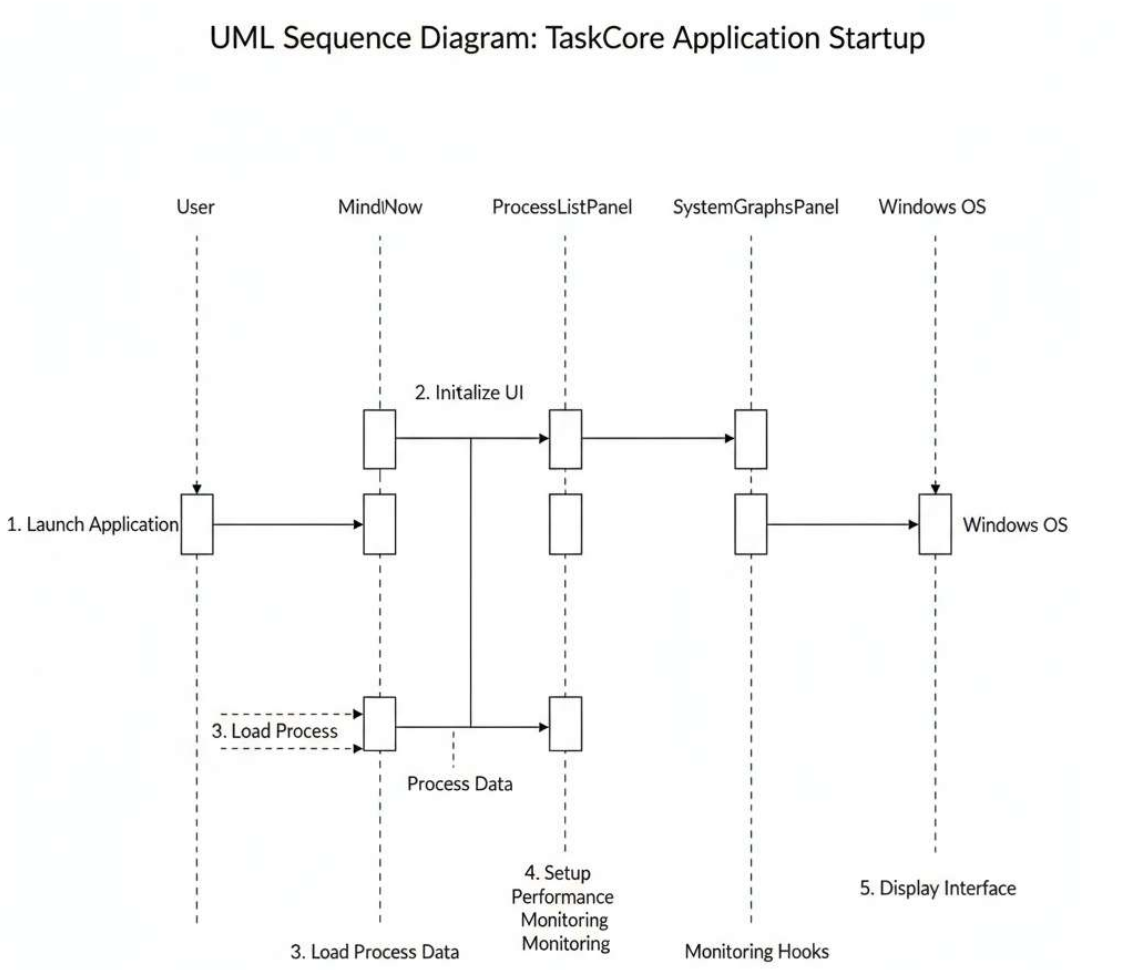
8.1 Class Diagram



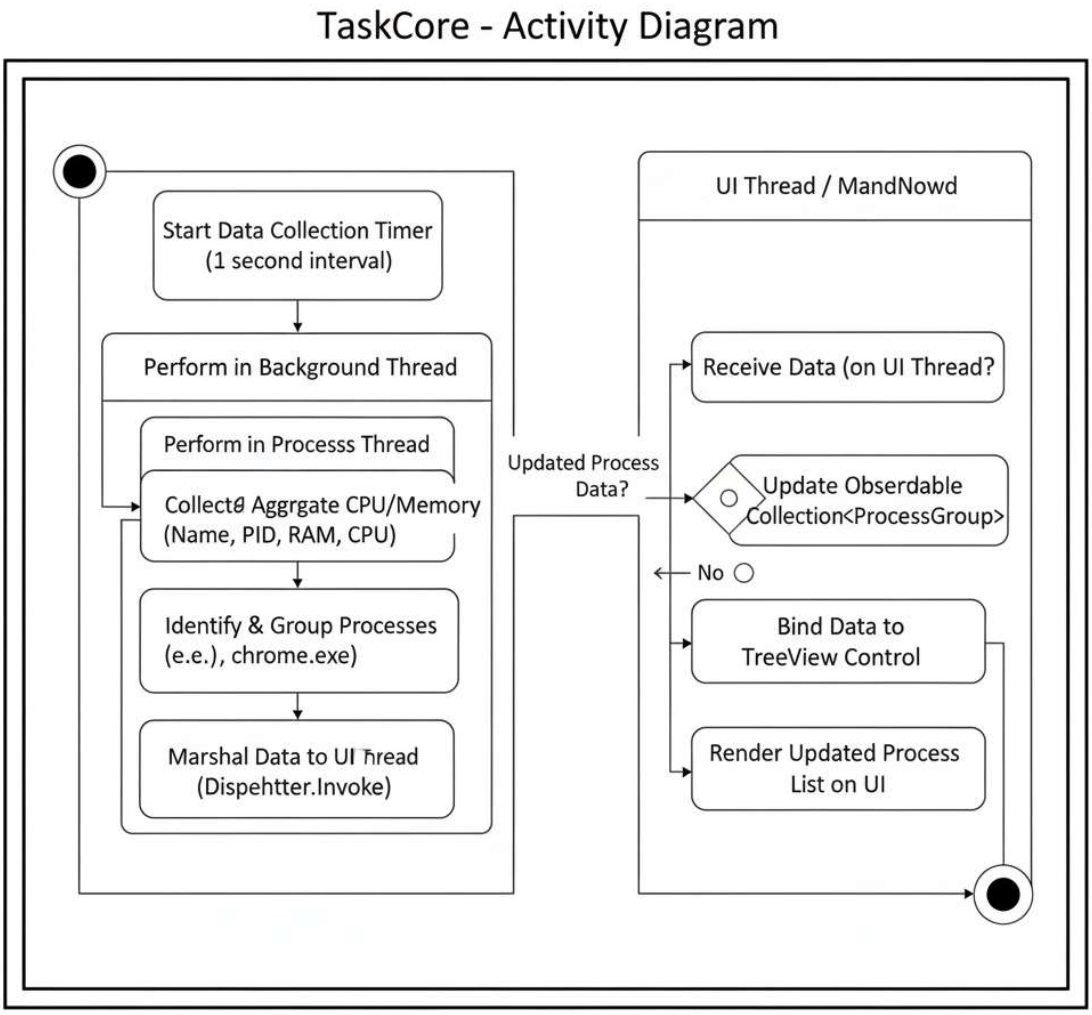
8.2 Use – Case Diagram



8.3 Sequence Diagram

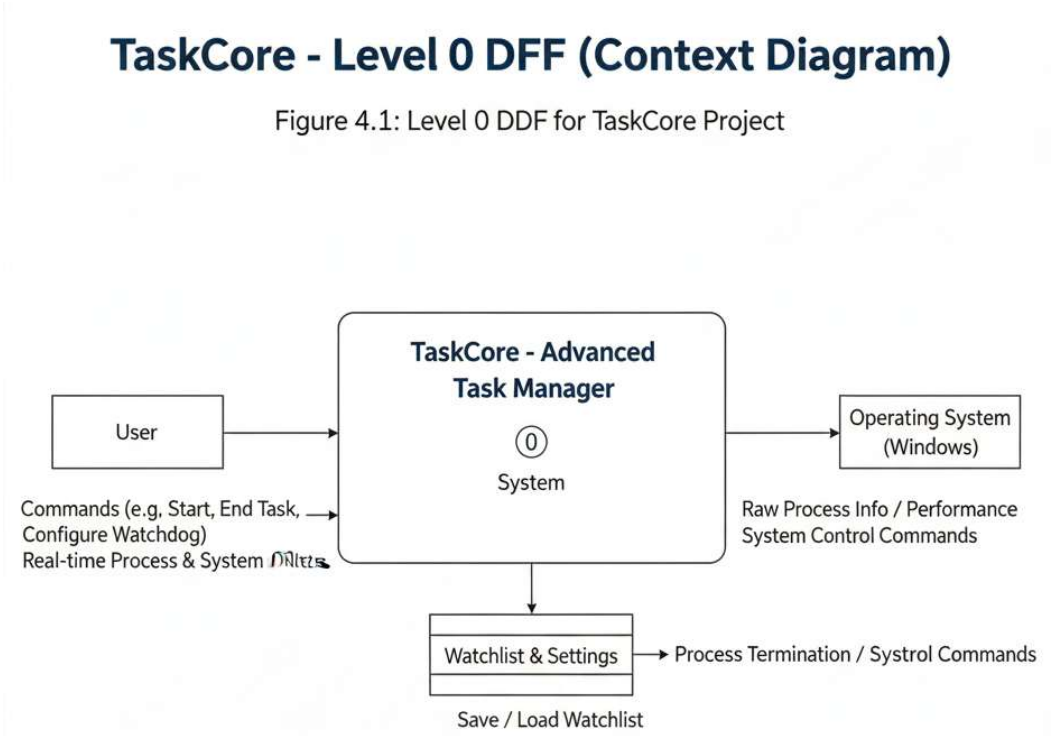


8.4 Activity Diagram

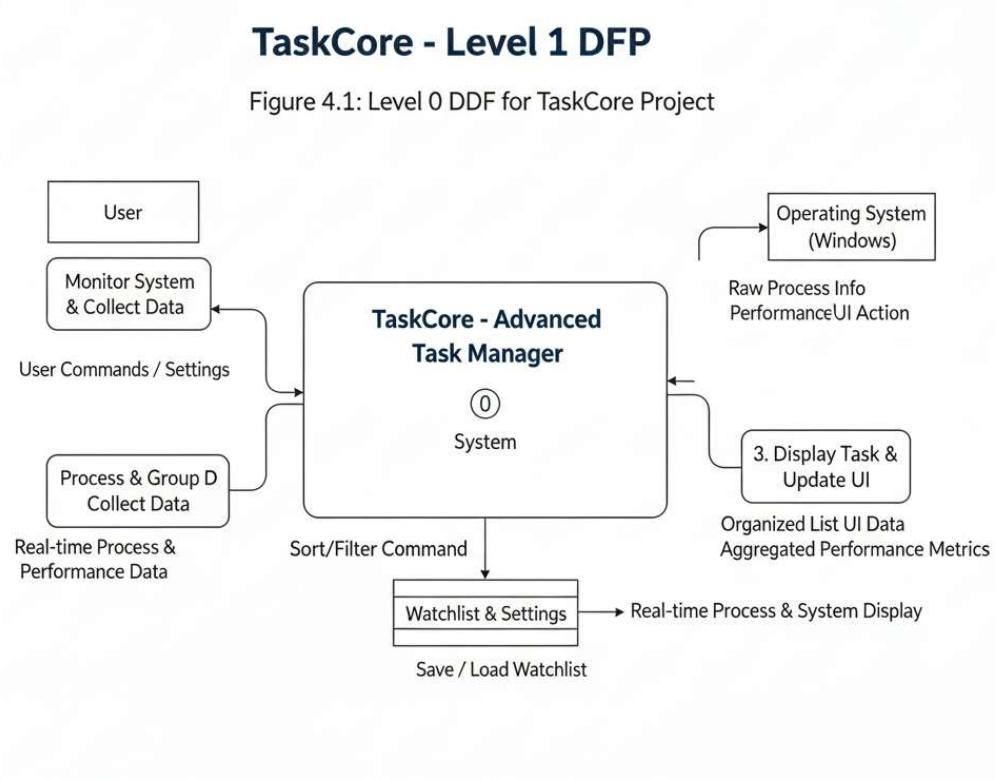


8.5 Data – Flow Diagram

- **Level Zero (0)**



- **Level One (1)**



- **Level Two (2)**

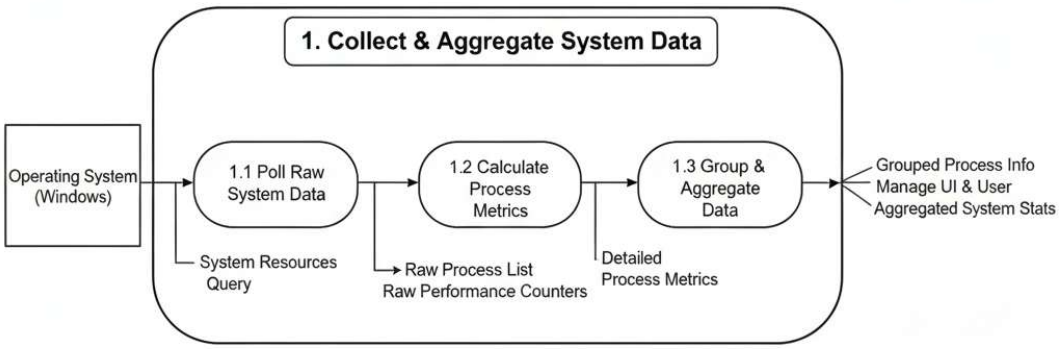


Figure 4.2: Level 2 DDF for TaskCore Project - Collect & Aggregate System Data

Chapter 9: Limitations & Future Enhancements

9.1 Limitations

Despite its comprehensive feature set, the TaskCore system has the following limitations:

1. **Platform Dependency**: TaskCore is built on the .NET Framework and WPF, making it strictly compatible with Windows 10 and Windows 11. It cannot run on non-Windows platforms or on earlier Windows versions without modifications.

2. **Administrative Privileges Requirement:** Core functionality such as process termination and WatchDog enforcement relies on UAC elevation. In environments with restrictive UAC policies or where users lack administrative rights, some features may not operate.

3. **Resource Overhead:** Although optimized for under 200 MB memory usage, TaskCore still consumes noticeable CPU and memory when monitoring hundreds of processes and rendering live charts, which may impact performance on low-end hardware.

4. **Limited Historical Data Storage:** Performance metrics history is stored in memory during runtime only. Upon application restart, all historical data is lost. There is no persistent storage of long-term performance trends.

5. **No Auto-Update Mechanism:** TaskCore does not include an integrated update facility. Users must manually download and install new versions, which may lead to outdated deployments.

6. **Basic Security Model:** The WatchDog module uses process name and path blacklisting only. It lacks more advanced threat detection methods such as behavior analysis, machine-learning-based anomaly detection, or integration with external security databases.

7. **Single-User Design:** Designed for local, single-user operation without support for multi-user environments or centralized configuration management, limiting its use in enterprise or network-wide deployments.

8. **UI Customization Constraints:** While users can adjust refresh intervals and graph settings, deeper UI customization (themes, layout rearrangement) is limited. Advanced personalization requires code modifications.

9.2 Future Enhancement

The following enhancements can address current limitations and expand TaskCore's capabilities:

1. **Cross-Platform Support:** Migrate to .NET Core/.NET 5+ with cross-platform UI frameworks (e.g., Avalonia) to support Linux and macOS, broadening the user base.

2. **Persistent Data Storage:** Implement a lightweight embedded database (e.g., SQLite) to store historical performance data, security logs, and user settings across sessions for long-term analysis.

3. **Automated Update Mechanism:** Integrate an auto-update service to check for and apply new versions automatically, ensuring users always run the latest, most secure release.

4. **Enhanced Security Monitoring**
   Augment the WatchDog module with:
   - Behavior-based anomaly detection
   - Integration with antivirus/EDR APIs
   - Machine-learning-driven threat classification
   - Real-time alerts via email or messaging platforms

5. **Role-Based Access Control:** Introduce user authentication and role-based permissions to enable supervised multi-user deployments. Administrators could delegate limited access to support staff.

6. **Plugin Architecture:** Redesign core modules to support plugins, allowing third-party developers to extend functionality (e.g., network monitoring, disk I/O analysis, custom alerts).

7. **Advanced UI Customization:** Expose theme, layout, and panel configuration settings through a user-friendly settings editor. Enable drag-and-drop panel arrangements and custom dashboard creation.

8. **Remote Monitoring and Management:** Add a client-server component to enable remote monitoring of multiple machines from a centralized console, supporting enterprise-scale system management.

9. **Performance Optimization Algorithms:** Optimize data sampling and rendering pipelines using more efficient data structures and multithreading to further reduce CPU and memory overhead.

10. **Integration with Cloud Services:** Provide optional integration with cloud platforms (e.g., Azure Monitor, AWS CloudWatch) for aggregated metrics, alerting, and long-term trend analysis in enterprise environments.

Chapter 10: Conclusion

The TaskCore system management application successfully addresses the need for a unified, efficient solution to Windows system administration tasks. By integrating process management, performance monitoring, security enforcement, and memory optimization into a single application, TaskCore eliminates the complexity of managing multiple separate tools.

The project achieved its primary objectives of delivering a professional, user-friendly interface with comprehensive functionality while maintaining resource efficiency under 200MB memory usage. Key features include real-time process visualization with TreeView hierarchy, dynamic performance graphs using LiveCharts2, automated security monitoring through the WatchDog module, and intelligent memory cleanup capabilities.

TaskCore's modular architecture and adherence to established design patterns ensure maintainability and extensibility. The application successfully demonstrates how modern desktop development techniques can create powerful system management tools that are both accessible to general users and capable of meeting advanced administrative requirements.

While current limitations include Windows-only compatibility and lack of persistent data storage, the foundation established provides a solid platform for future enhancements including cross-platform support, advanced analytics, and enterprise features.

TaskCore represents a significant step forward in desktop system management applications, providing users with the tools needed to effectively monitor, secure, and optimize their computing environment through a single, cohesive interface.

Chapter 11: Appendices

11.1 Business Model

11.1.1 Product Overview

TaskCore is designed as an open-source desktop system management application with potential for future commercialization through enterprise features and support services.

Current Model: Open-Source Foundation

- Free distribution under MIT License
- Source code available on GitHub/GitLab
- Community-driven development and contributions
- Educational and personal use focus

Future Commercial Opportunities:

- **TaskCore Enterprise**: Advanced features for business environments
- **Premium Support**: Professional technical support and training
- **Custom Deployment**: Tailored implementations for large organizations
- **Plugin Marketplace**: Third-party extensions and integrations

11.1.2 Target Market Segments

Primary Market:

- **System Administrators**: IT professionals managing Windows environments
- **Power Users**: Technical enthusiasts requiring advanced system control
- **IT Support Teams**: Help desk and technical support personnel
- **Educational Institutions**: Computer science students and instructors

Secondary Market:

- **Small-Medium Enterprises**: Businesses needing cost-effective system monitoring
- **Independent Developers**: Software developers optimizing development environments
- **Cybersecurity Professionals**: Security analysts requiring process monitoring tools

11.1.3 Value Proposition

Core Benefits:

- **Unified Interface**: Single application replacing multiple system tools
- **Cost-Effective**: Free alternative to expensive enterprise monitoring solutions
- **Resource Efficient**: Optimized performance with minimal system impact
- **Security-Focused**: Built-in threat detection and process blocking capabilities
- **User-Friendly**: Professional interface accessible to various skill levels

Competitive Advantages:

- **Integration**: Combined functionality vs. fragmented tool ecosystem
- **Performance**: Optimized resource usage under 200MB memory
- **Security**: Proactive WatchDog monitoring vs. reactive tools
- **Customization**: Modular architecture supporting extensibility

11.2 Product Development Details

11.2.1 Technical Architecture

Development Framework:

- **Platform**: Microsoft .NET Framework 4.7.2+
- **UI Technology**: Windows Presentation Foundation (WPF)
- **Programming Language**: C# 8.0+
- **IDE**: Microsoft Visual Studio 2019/2022

Key Dependencies:

```
// Core WPF and System APIs

using System.Diagnostics;          // Process management
using System.Windows;              // WPF framework
using System.ComponentModel;       // Data binding

// Third-party Libraries

LiveCharts2                  // Performance visualization
Newtonsoft.Json              // Configuration serialization
```

11.2.2 Module Implementation Details

Process List Module (ProcessListPanel.cs):

- **Core Class**: ProcessListPanel : UserControl
- **Key Components**:
  - TreeView processTreeView - Hierarchical process display
  - ToggleButton headers - Sortable column headers (Name, PID, Memory, CPU)
  - Button Btn_end_task - Process termination control
- **Data Binding**: ObservableCollection<ProcessInfo> for real-time updates
- **Sorting Logic**: Multi-column sorting with ascending/descending indicators

Main Application Controller (MainWindow.xaml.cs):

- **Core Class**: MainWindow : Window, INotifyPropertyChanged
- **Key Properties**:
  - ObservableCollection<ProcessInfo> Processes - Process data collection
  - bool IsSidebarOpen - Navigation state management
  - object CurrentPanelView - Active module display
- **Timer Management**: DispatcherTimer _refreshTimer (1-second intervals)
- **Process Operations**: Integration with Windows Process API

11.3 API and Web Services Details

11.3.1 Windows System APIs

TaskCore integrates with core Windows APIs for system-level operations:

Process Management APIs:

```
// System.Diagnostics namespace

Process.GetProcesses()          // Enumerate running processes
Process.GetProcessById(int id)  // Retrieve specific process
process.Kill()                  // Terminate process
process.WorkingSet64            // Memory usage information
```

Performance Counter APIs:

```
// System.Diagnostics.PerformanceCounter

PerformanceCounter cpuCounter    // CPU usage monitoring
PerformanceCounter memoryCounter // Memory usage tracking
```

Windows Management Instrumentation (WMI):

```
// Enhanced process information

ManagementObjectSearcher processorQuery
ManagementObjectSearcher memoryQuery
```

11.3.2 Internal Service Architecture

Data Collection Services:

- **ProcessMonitoringService**: Background process enumeration and monitoring
- **PerformanceDataService**: CPU and memory metrics collection
- **SecurityMonitoringService**: WatchDog threat detection and response
- **MemoryOptimizationService**: RAM cleanup and optimization procedures

Configuration Services:

- **ConfigurationManager**: Settings persistence using JSON serialization
- **SecurityPolicyManager**: WatchDog blacklist management and storage
- **UserPreferencesService**: UI customization and preference management

11.3.3 Future API Integration Plans

Planned External Integrations:

- **Windows Event Log API**: Enhanced security event logging
- **Windows Performance Toolkit**: Advanced performance analysis
- **Microsoft Graph API**: Enterprise integration for Office 365 environments
- **Azure Monitor API**: Cloud-based metrics aggregation and alerting

Third-Party Service Integration:

- **Antivirus API Integration**: Enhanced threat detection capabilities
- **SIEM Platform APIs**: Security information and event management
- **Network Monitoring APIs**: Expanded system monitoring coverage
- **Cloud Storage APIs**: Configuration backup and synchronization

11.3.4 Security and Authentication

Current Security Model:

- **UAC Integration**: User Account Control for privilege escalation
- **Process Validation**: Critical system process protection

- **Secure Configuration**: Protected settings storage with access control

Future Authentication Enhancements:

- **Active Directory Integration**: Enterprise user authentication
- **Certificate-Based Security**: Digital signature validation
- **Multi-Factor Authentication**: Enhanced security for enterprise deployments
- **Audit Logging**: Comprehensive security event tracking and reporting

11.3.5 Performance Optimization

API Call Optimization:

- **Caching Strategies**: Intelligent process information caching
- **Batch Operations**: Grouped API calls for improved efficiency
- **Background Threading**: Non-blocking UI operations
- **Resource Management**: Automatic cleanup and memory optimization

Monitoring and Metrics:

- **Performance Counters**: Internal application performance tracking
- **Resource Usage Monitoring**: CPU and memory consumption analysis
- **API Response Times**: System call performance measurement
- **Error Rate Tracking**: API failure monitoring and recovery

Bibliography

1. **Microsoft Documentation**

   - Microsoft Docs: Process Class. docs.microsoft.com/en-us/dotnet/api/system.diagnostics.process
   - Windows Presentation Foundation (WPF) Guide. docs.microsoft.com/en-us/dotnet/desktop/wpf/
   - .NET Framework Guide. docs.microsoft.com/en-us/dotnet/framework/

2. **Books**

   - Troelsen, A., & Japikse, P. (2021). *Pro C# 9 with .NET 5*. Apress.
   - Albahari, J., & Albahari, B. (2021). *C# 9.0 in a Nutshell*. O'Reilly Media.

3. **Articles & Academic Papers**

   - Basili, V. R., & Rombach, H. D. (1988). "The TAME Project: Towards Improvement-Oriented Software Environments." *IEEE Transactions on Software Engineering*.
   - "Measuring Software Design Complexity." Card, D. N., & Agresti, W. W. *The Journal of Systems and Software*.

4. **Open-Source Libraries**

   - LiveCharts2. https://github.com/beto-rodriguez/LiveCharts2
   - Newtonsoft.Json. https://www.newtonsoft.com/json

5. **Online References**

   - GeeksforGeeks: Logical Data Flow Diagrams. https://www.geeksforgeeks.org/logical-data-flow-diagram-dfd/
   - ScienceBuddies: How to Write a Bibliography. https://www.sciencebuddies.org/science-fair-projects/science-fair/writing-a-bibliography-apa-format

6. **Stack Overflow and Community Forums**

   - "Process Tree and Hierarchical Process Visualization in WPF." Stack Overflow threads and MSDN communities.

7. **Your Own Source Code and Documentation**

   - TaskCore internal design notes, code comments, and specifications (unpublished).