SRS

Project Overview and Benefits

In modern computing environments, system changes introduced by third-party executables and scripts are both common and consequential. These modifications affect files, folders, and registry entries, often without transparency, especially when the underlying code is closed-source. Existing monitoring tools such as Microsoft Sysinternals' Process Monitor or various open-source alternatives demonstrate the importance of tracking system changes, but they frequently fall short in two key areas: accessibility and portability. Many require expert knowledge to interpret low-level outputs, and most are tied to specific operating system versions, limiting their use in specialized environments such as Windows Embedded or Windows IoT.

Academic research in digital forensics and system auditing highlights the critical role of capturing registry and file system changes in supporting incident response, malware detection, and compliance audits. Yet, there remains a gap between highly technical forensic-grade tools and practical, user-friendly solutions for everyday use.

WinChangeMonitor aims to fill this gap by combining the rigor of forensic monitoring with an intuitive interface that broadens accessibility for both technical and non-technical users. Unlike existing tools, it emphasizes:

- Ease of use through clear, human-readable reporting.
- Cross-version compatibility, including specialized Windows environments.
- Actionable insight, reducing guesswork and risk when running unverified or legacy software.

By addressing these shortcomings, WinChangeMonitor positions itself not just as another monitoring tool, but as a practical bridge between complex system auditing solutions and the needs of everyday developers, security analysts, and administrators.

Project Scope

The WinChangeMonitor project's scope includes designing, creating, and delivering a Windows-based monitoring tool that can track and report on file system and registry changes brought about by third-party scripts and executables. The goal of this project is to offer a workable solution that blends an intuitive user interface with the technical precision of forensic-grade monitoring.

Within the scope of this project, the team will develop a WinForms application using C# and the .NET Framework 4.8 as the core programming environment. The system will support baseline and post-execution inventories to capture registry entries, file attributes, and directory structures, followed by comparison and difference analysis to highlight modifications. Windows APIs will be leveraged to provide the necessary low-level system access, while reporting features will be designed to present results in a clear, human-readable format with options for structured export in JSON or XML. Configuration management will be maintained through GitHub to ensure effective version control and team collaboration, and the development process will be supported by CI/CD practices using Jenkins, Docker, and GitHub to automate builds, testing, and deployment. An optional SQL database may also be integrated to store system inventory data in larger-scale environments.

The project will not expand into advanced forensic analysis, integration with enterprise-scale SIEM solutions, real-time malware detection and remediation, or platform support beyond the Windows family

of operating systems. These areas are considered out of scope in order to maintain focus on the project's objectives. A Minimum Viable Product that can record inventory, analyse differences, and report will be one of the project's deliverables. Other deliverables will be exportable reports for documentation and auditing needs. The project will be completed with a final release version of the application that includes installation instructions and user documentation.

By defining these boundaries, the project ensures that WinChangeMonitor will deliver a reliable and usable tool that supports developers, system administrators, and security analysts in managing risks associated with third-party executables, while staying aligned with the available time, resources, and technical constraints.

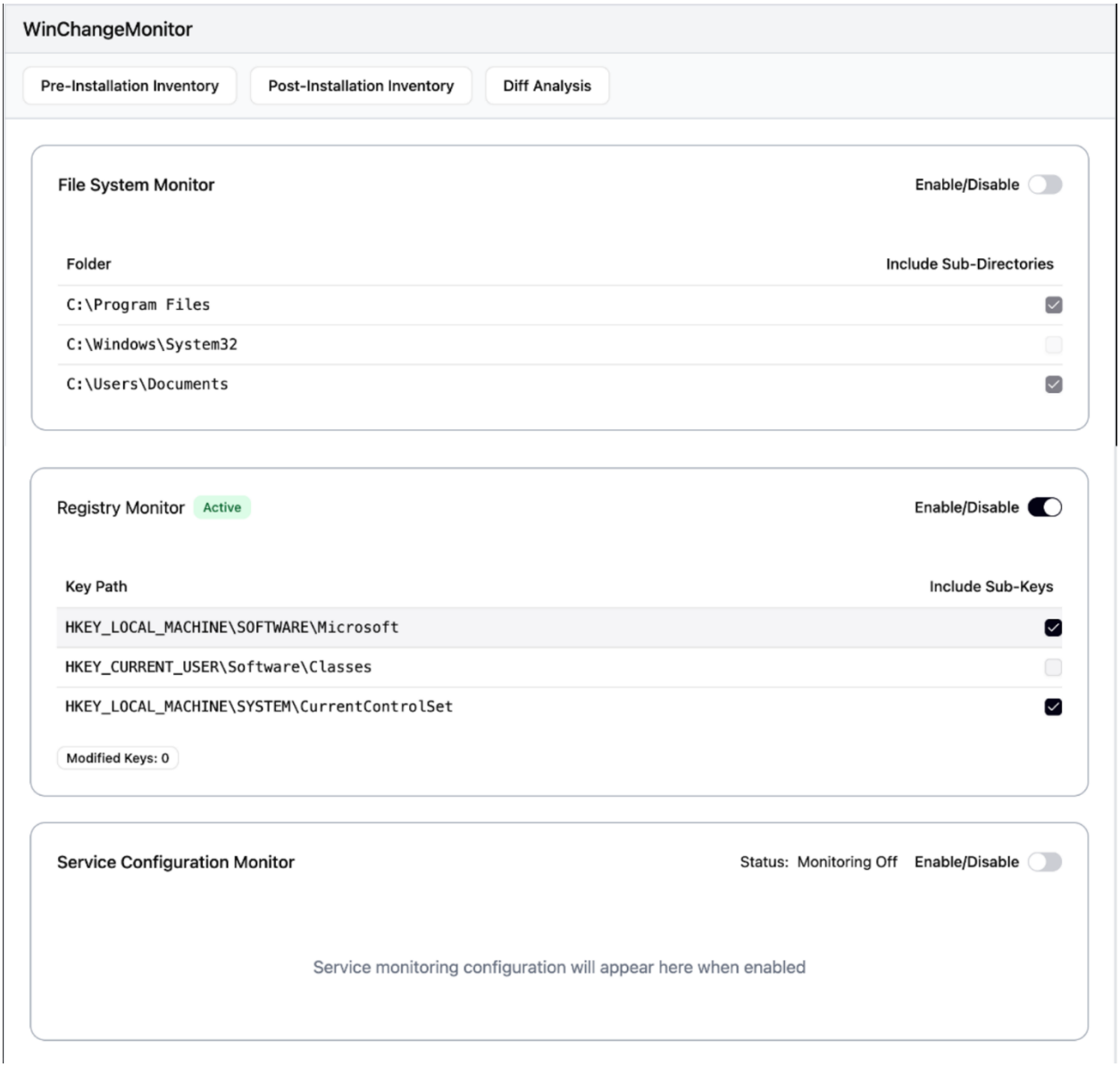
High-level requirements

The table below corresponds to a high-level requirements list outlining both the functional and non-functional requirements for the WinChangeMonitor project. Each requirement is paired with its method of evaluation or acceptance test, along with the user story and assigned priority level for "wish" requirements. D/W makes a distinction between Design (Need) requirements and Wish requirements in the table below. Wish requirements are regarded as non-essential improvements, whereas design requirements stand for fundamental system needs. Wish requirements may be implemented in subsequent sprints if time and resources allow, but they are meant to be addressed after the Minimum Viable Product (MVP) is completed.

Group 3	**REQUIREMENTS LIST**			**Issued on: 9/28/2025**
D/W	**Wt (1-5, 1 = Lowest Priority)**	**Requirement**	**User Story**	**Method of Evaluation / Acceptance Test**
	Functional Requirements			
D	N/A	Enable/disable monitoring for File System, Registry, and Service configuration	The user shall be able to selectively enable or disable monitoring for the File System, Registry, and Service configuration.	Verify UI toggles exist and correctly activate/deactivate monitoring; run tests with each option enabled/disabled.
D	N/A	Select subset of drives/folders for File System monitoring	The user shall be able to specify a subset of drives or folders to track when File System monitoring is enabled.	Configure subset of drives/folders and verify only those are logged during execution.
D	N/A	Select subset of registry keys for Registry monitoring	The user shall be able to specify a subset of registry keys to track when Registry monitoring is enabled.	Configure registry subset and confirm changes outside that set are not logged.
D	N/A	Perform pre-installation system inventory via UI	The user shall be able to initiate a pre-installation system inventory through the UI.	Attempt pre-install inventory, confirm system captures baseline, block if no monitoring enabled.
D	N/A	Display UI status during inventory execution	The user shall be able to view a status indicator on the UI showing that the program is performing inventories and has not stalled or crashed.	Simulate long-running inventory and confirm progress/status indicator updates and system is responsive.
D	N/A	Allow system reboot between execution and post-installation analysis	The user shall be able to reboot the system after executing third-party software and before performing the post-installation inventory and difference analysis.	Perform baseline inventory, reboot, run software and confirm baseline inventory persists across reboot.
D	N/A	Perform post-installation inventory and difference analysis	The user shall be able to initiate a post-installation system inventory and difference analysis through the UI.	Run baseline, execute third-party program, run post-install analysis; confirm diffs are logged.

D	N/A	Generate HTML difference report	The user shall be able to receive a system difference report in HTML format.	Verify HTML report is generated with correct entries; validate file integrity.
W	4	Export reports as PDF, JSON, etc.	The user shall be able to export system difference reports in formats such as PDF or JSON.	Trigger export; validate correct formats generated.
D	N/A	Show counts of modified files by directory	The user shall be able to view the number of modified files within each directory.	Execute monitored program, confirm counts displayed correctly in UI.
D	N/A	Show new, modified, deleted files separately in UI with color distinction	The user shall be able to view newly added, modified, and deleted files in separate sections, visually distinguished in the UI.	Trigger test with file changes; confirm correct color categorization.
W	3	Provide filtering/search within reports	The user shall be able to filter and search for specific changes within generated reports.	Trigger test with file changes; confirm correct color categorization.
W	3	Support saving/loading monitoring configurations	The user shall be able to save and reload monitoring configurations for reuse.	Save configuration, reload later, verify settings persist correctly.
	Non-Functional Requirements			
D	N/A	Interface must be intuitive for technical and non-technical users	The user shall be able to interact with an intuitive interface that is usable by both technical and non-technical users.	Conduct user testing; survey ease of navigation; measure time-to-completion for inventory tasks.
D	N/A	Reports must be accurate and consistent across runs	The user shall be able to rely on reports that are accurate and consistent across repeated runs.	Compare logs across repeated runs of same executable; confirm consistent results.
D	N/A	Must support multiple Windows versions (Win10, Win11, Embedded, IoT)	The user shall be able to run the tool across multiple versions of Windows, including specialized environments such as Windows Embedded or Windows IoT.	Run on each supported version; verify monitoring and reporting features function identically.
D	N/A	Reports must avoid exposing sensitive system data such as user credentials, authentication tokens, or secure registry values.	The user shall be able to generate reports that omit or anonymize sensitive information, including user credentials, authentication tokens, and secure registry values.	Run test cases with known sensitive data (e.g., passwords in registry keys, user profile data) and confirm these values are either excluded or replaced with anonymized placeholders in the report.
W	3	Exported reports should be viewable on any system without the tool installed	The user shall be able to view exported reports on any system without requiring the WinChangeMonitor tool to be installed.	Open exports on a clean system; verify readability without additional software.
W	3	Codebase should follow documented standards and versioning practices	The user shall be able to benefit from a codebase that follows documented standards and versioning practices, ensuring long-term support.	Conduct peer code reviews; confirm compliance with contribution guidelines.
W	2	Optional database integration (SQL) for larger inventories	The user shall be able to run the tool with optional database integration (SQL) to handle large-scale inventories.	Populate with >1M entries; confirm database mode runs without crash.
W	3	Color distinctions in UI must be perceivable by colorblind users	The user shall be able to perceive UI distinctions (such as color coding) even if colorblind or visually impaired.	Verify UI passes WCAG contrast standards; offer alternate icons/labels.
W	2	Key user and system actions, such as initiating inventories, generating reports, exporting files, and configuration changes, should be logged for troubleshooting and auditing.	The user shall be able to review logs of critical actions performed by the tool, including inventories, report generation, exports, and configuration changes.	Perform these actions during testing; verify log files include accurate entries with timestamps and identifiers for each action.

High-Level Interface Design]
The figure below is the high-level interface design for this project.



Appendix
Team Contributions

Jeff Rose: I created the initial revision for the Requirements as User Stories which we all then worked collaboratively on to improve and expand upon. In addition, I offered feedback on the UI design as well as a suggestion on how we can tackle support for application resizing to allow the UI to remain usable through all supported sizes.

Anjian Chen: We all discussed about the SRS report during our meeting after lecture, where I proposed that the application should have a status bar when scanning files, or display how many files have been scanned. Also, I suggested that there should be a separate window (apart from the main menu) for the users to adjust which certain files/directories the users want to track on. I've also recorded the meeting notes after our meeting on Tuesday for references. Later, I came up with two more requirements regarding when users are looking through the differences after the scanning has completed. In addition, I suggested some UI adjustment to Yeryoung's UI prototype.

Yu Wu: In preparation for our UI design discussion, I focused on window adjustability as a usability concern. I suggested that the requirement clearly state, “The window must be adjustable.” From there, I drafted a user story: “The user shall be able to adjust the size of the window vertically and horizontally.” I also defined an acceptance criterion: “Conduct a test run on Windows OS, and the window should be adjustable.” This contribution ensures our design accommodates user expectations and aligns with practical testing.

Yeryoung Kim: I participated in stand-up meetings and contributed to drafting the UI design in Figma, basing it on the draft requirements and incorporating feedback from team members. I also reviewed the SRS submission and provided several suggestions to improve the document.

Princely Oseji: Collaborated with the team to create the project proposal, combined SCMP/SPMP and the SRS documents. Created the initial drafts for these documents based on discussions during the stand-up meetings and modified them according to feedback given by both the team and the professor.