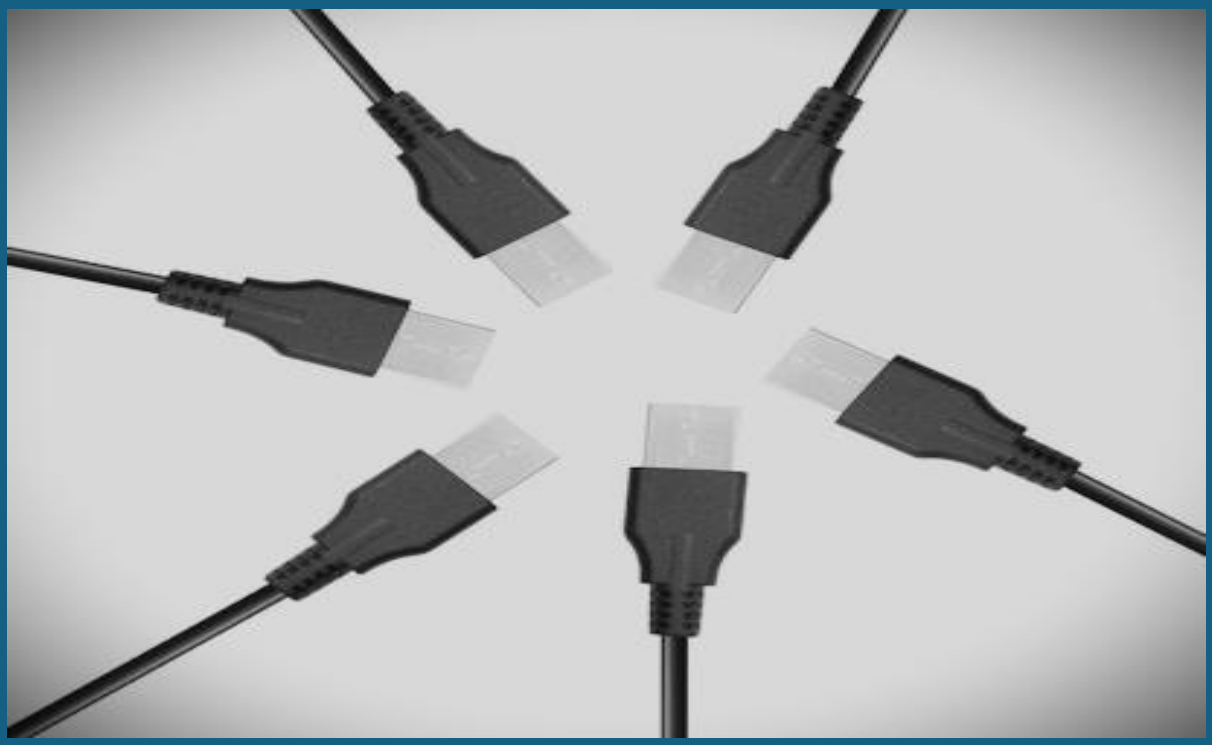
USB SENTINEL

GROUP 37 Documentation

Software Design 2
(SDN260S)



A.QOTYWA 241131812
M.NYATI 240056779
LM.NCANAZO 240979044
S.DLAMINI 240589858

Department of Electrical, Electronic and
Computer Engineering

Contents

1. Problem Definition and Requirements

Modern organisations rely heavily on USB devices for transferring data and connecting peripherals. Unauthorised use of USB connections pose significant risks including theft of data as well as malware attacks.

This project looks to build a desktop application that detects, documents, and categorises the activities of USB devices in real time. This application can be used to monitor both administrators and users to ensure safety as well as auditing.

The system must detect when a device is connected or disconnected. It should record detailed information like device name, manufacturer, vendor ID, and type. It will distinguish between different device categories, such as keyboards, mice, controllers, and phones. The tool should support both automatic and manual logging modes and keep a continuously updated log file (usb_log.txt) with timestamps. A graphical interface should let users view detected devices, apply filters, and test the detection feature.

Detection accuracy is achieved by using two mutually complementary techniques, namely LibUSB, accessed over usb4java, as well as Windows Management Instrumentation (WMI). LibUSB offers low-level information on USB interaction, whereas WMI offers detailed information on storage devices such as size as well as serial numbers. Combining the two techniques assures extensive coverage both of mass storage devices as well as peripherals on Windows platforms.

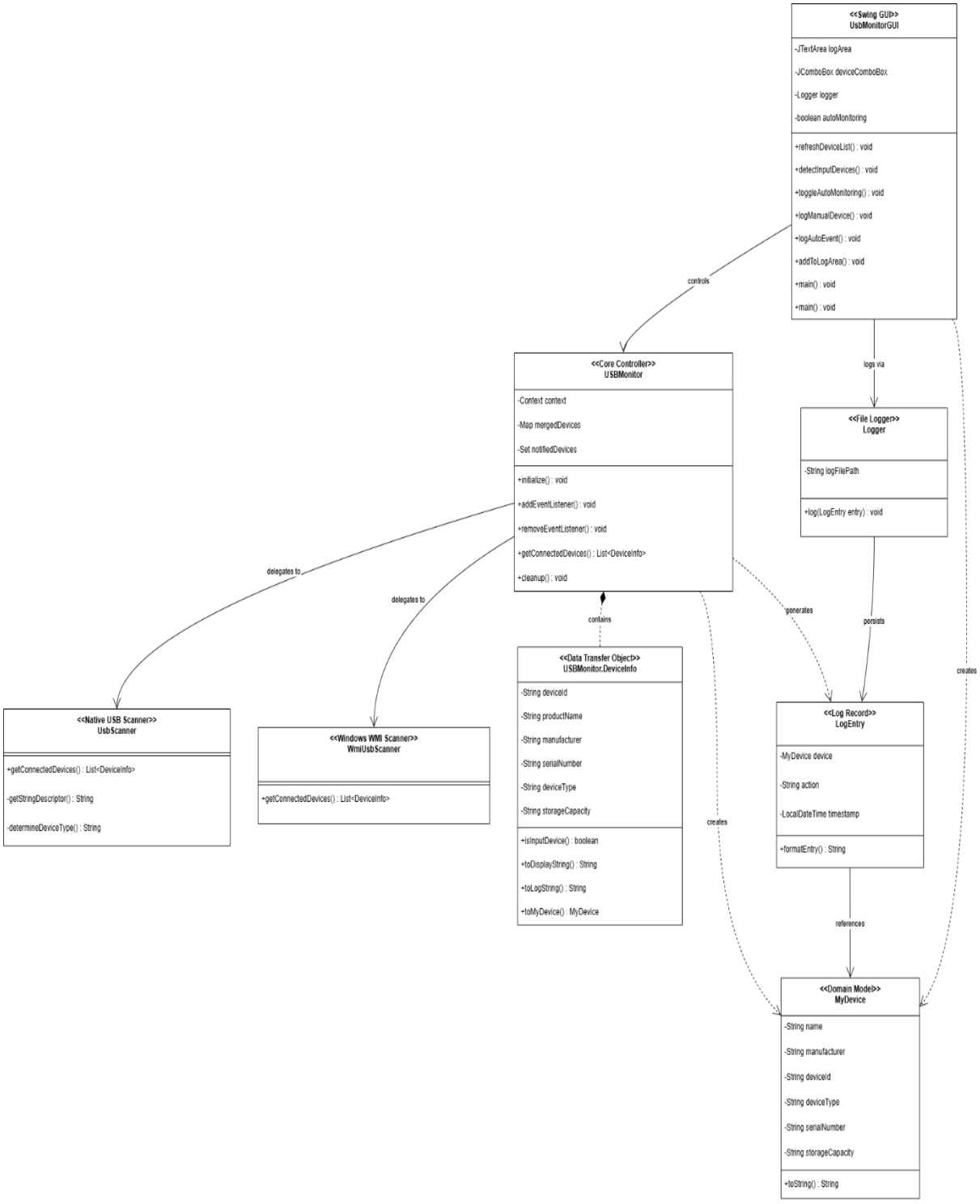
2. System Architecture and Design

2.1 Overview

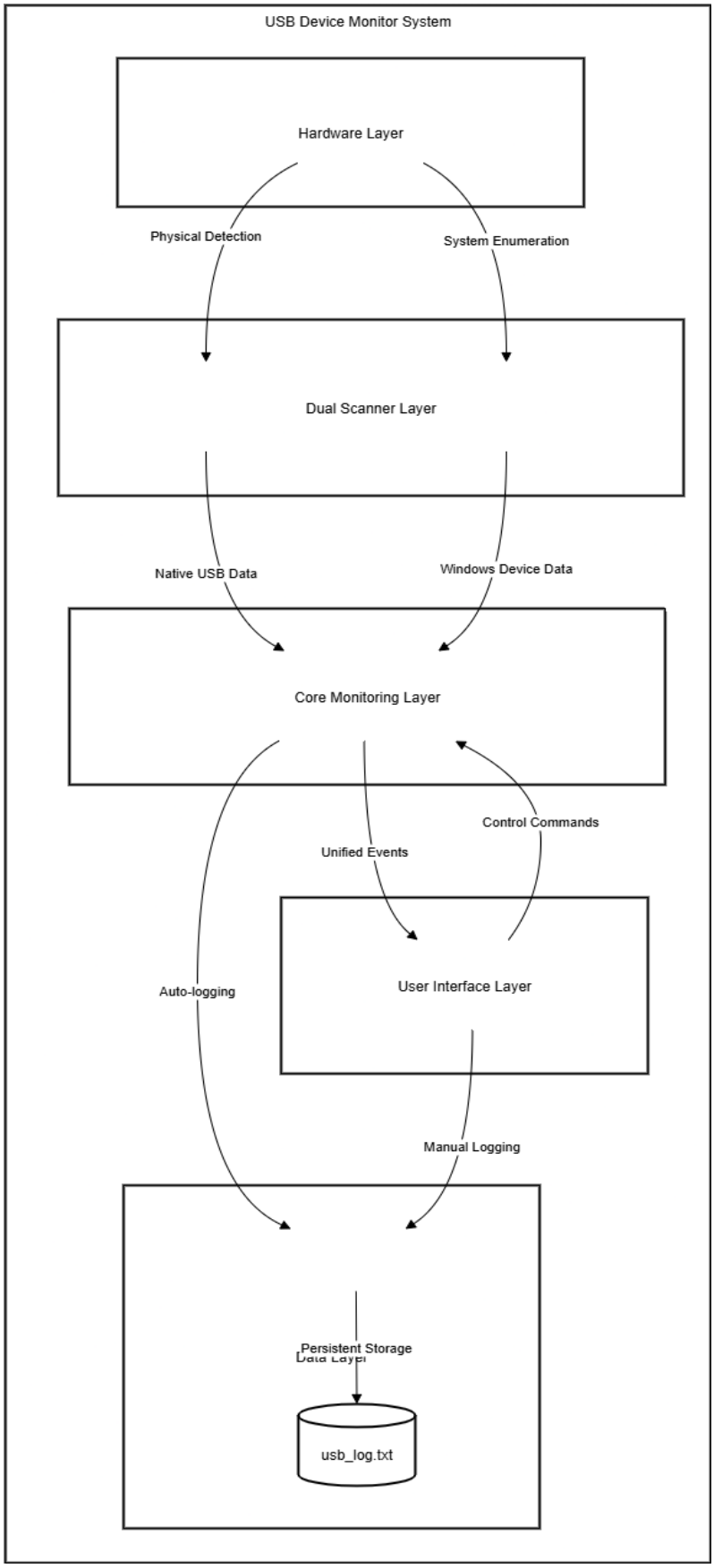
The framework uses usb4java (LibUSB) to capture hardware-level USB events. There is a polling subsystem that routinely searches for device changes and generates events due to device disconnection or connection. The GUI, built in the form of Swing supports user interaction, manual logging and device visualisation.

The combined architecture unifies LibUSB and WMI scanner data in the USBMonitor class. A background thread, every three seconds scans for connected devices which combines the results and creates events on detection of actual disconnections or connections. This combined design minimises the impact on system resources with efficient monitoring.

2.2 UML Class Diagram



2.3 System Architecture Diagram



2.4 Development Environment and Tools

Component	Tool / Version
Language	Java SE 17
IDE	Visual Studio Code
External Library	usb4java v1.3.0
Native Driver	libusb-1.0.dll
Operating System	Windows 10 (64-bit)
GUI Toolkit	Java Swing / AWT
Dual Detection Modules	LibUSB (usb4java) + WMI integration

3.1 Object-Oriented Design

The code is made up of modular Java Classes:

MyDevice: A class representing the USB device with name, manufacturer, ID, type.

LogEntry: A timestamped log of the device action (insert/remove).

Logger: Manages file writing for all logged messages.

USBMonitor: Core hardware interface with LibUsb; checks devices, observes insertions/removals, and handles listeners.

UsbMonitorGUI: Swing-based GUI that brings all the modules together for user interaction.

WmiUsbScanner: Reads data from Windows Management Instrumentation, which stores capacity as well as serial number.

UsbScanner: Reads low-level USB information (vendor ID, product ID, manufacturer) using LibUSB.

Dual Polling Mechanism: Merges both scanners' output to a single list handled by USBMonitor.

3.2 Algorithms and Logic

Device Polling and Merge Algorithm

The Device Polling and Merge Algorithm works as follows: both scanners collect lists of associated devices, the lists being combined to suppress duplicates. The combined list is compared to the internal state. New ID triggers a "Device Connected" event, as a missing ID triggers a "Device Disconnected" event. This loop repeats every three seconds within a background thread, providing uninterrupted monitoring without putting the system under tension.

Device Type Identification

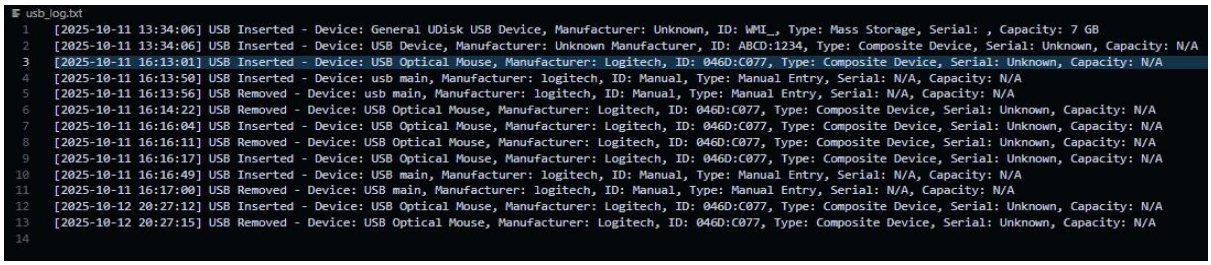
The device type is handled by the determineDeviceType() function, based on the class code of the USB class (such as Human Interface Devices = 0x03, Mass Storage = 0x08 and USB Hub = 0x09).

Input Device Detection

The input device is then recognised with the help of the keyword matching in product and type strings (e.g., "keyboard," "mouse," "controller").

Logging

Every event is written to the file usb_log.txt as follows:



GUI Interactions

The GUI supports both manual and automatic logging, device-type filter, along with controls to refresh, test detection, start/stop monitoring and clear logs.

4. Testing and Evaluation

Test Plan

Test Case	Description	Expected Result
TC1	Connects the USB flash drive	"Device connected" notification and log entry generated
TC2	USB device disconnection	"Device disconnected" notification displays and log
TC3	Connect the mouse	Detected as input device (mouse)
TC4	Toggle Auto Monitoring	Automatically monitors all device activities
TC5	Run on system without libusb	Shows error message and instructions
TC6	Device with missing product string	Displays fallback name (“USB Device 1234:5678”)
TC7	Device reconnects rapidly	Duplicate event ignored
TC8	Same device detected by both LibUSB and WMI	Single log entry produced
TC9	GUI under load	Remains responsive during background thread polling

Edge Case Testing

Edge cases were also considered. The devices with no product string are assigned fallbacks, the duplicate plug events are suppressed if the ID remains the same, and the GUI is always responsive.

5. Graduate Attributes Reflection

GA2 – Application of Scientific and Engineering Principles

The project illustrates the utilisation of software engineering as well as Java OOP principles. Abstraction and modularity are obtained through the separation of classes, whereas the execution in multithreaded form avoids GUI freezing during real-time polling. Persistent logging, device detection through algorithmic ways, as well as the use of external libraries (usb4java) to demonstrate the real-world utilisation of engineering concepts. Synchronisation as well as concurrency are tackled with the help of thread-safe collections like ConcurrentHashMap as well as CopyOnWriteArrayList.

GA7 – Sustainability and Impact of Engineering Activity

The system promotes sustainable computing by delivering efficient device monitoring with minimal CPU usage. It enhances security by preventing unauthorised device access, safeguards sensitive information and is designed to be maintainable and extensible, reducing technical debt. The dual detection approach increases reliability while maintaining negligible resource consumption.

6. Inter-Team Collaboration

The project was a team effort. Lukhanyo Ncanazo, as the Lead Developer, developed the USB detection logic, tied in the GUI, and did the testing. Malime Nyati, the Designer, made the UML diagrams and handled documentation. Sphesihle Dlamini, as Tester, did edge case validation as well as device logging accuracy. Ayabulela Dlamini, the Reviewer, polished the codebase and helped with the project presentation. This breakdown of tasks allowed efficient communication, proper task distribution, and easy transition from design to implementation.

7. Conclusion

USB Sentinel is a powerful desktop security product that uses LibUSB to undertake low-level communication as well as the use of WMI to gather system-level analysis. The tool classifies, detects, and tracks USB activity in real time through a combination of modular construction, effective logging and monitoring. Through combining hardware-level as well as system-level detection, the project delivers effective USB event tracking with enhanced responsiveness and stability, with no compromises on the specifications of the Software Design 2 brief, the SDN260S.

8. References

1. usb4java Documentation. http://usb4java.org
2. LibUSB API Reference. https://libusb.sourceforge.io/api-1.0/
3. Oracle. *Java SE 8 & 17 Documentation.* https://docs.oracle.com/javase/