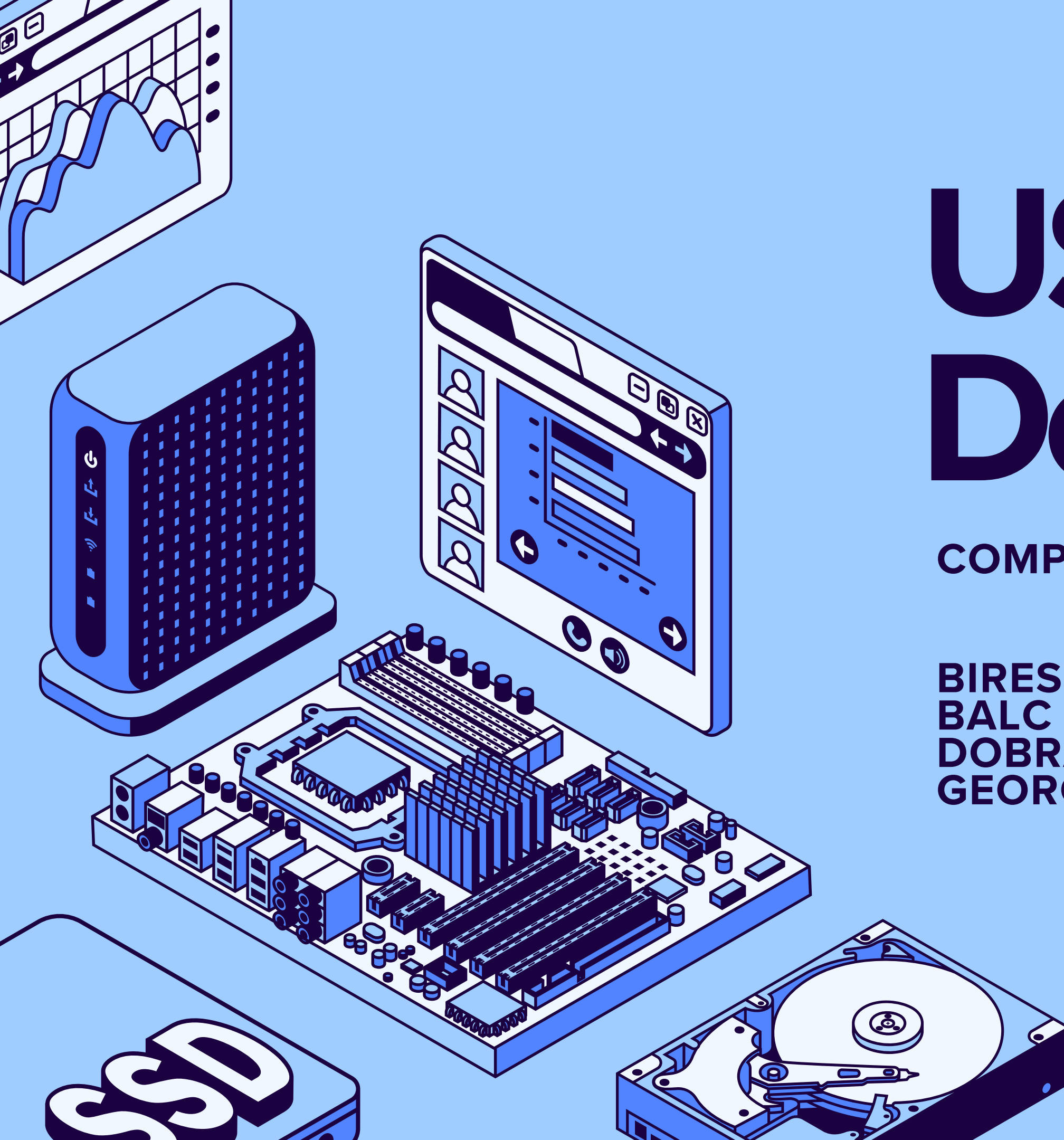


# USB Memory Detector

COMPUTER ENGINEER AND TESTING

BIRESCU IONUȚ  
BALC DRAGOS  
DOBRA DARIUS  
GEORGESCU RAREȘ

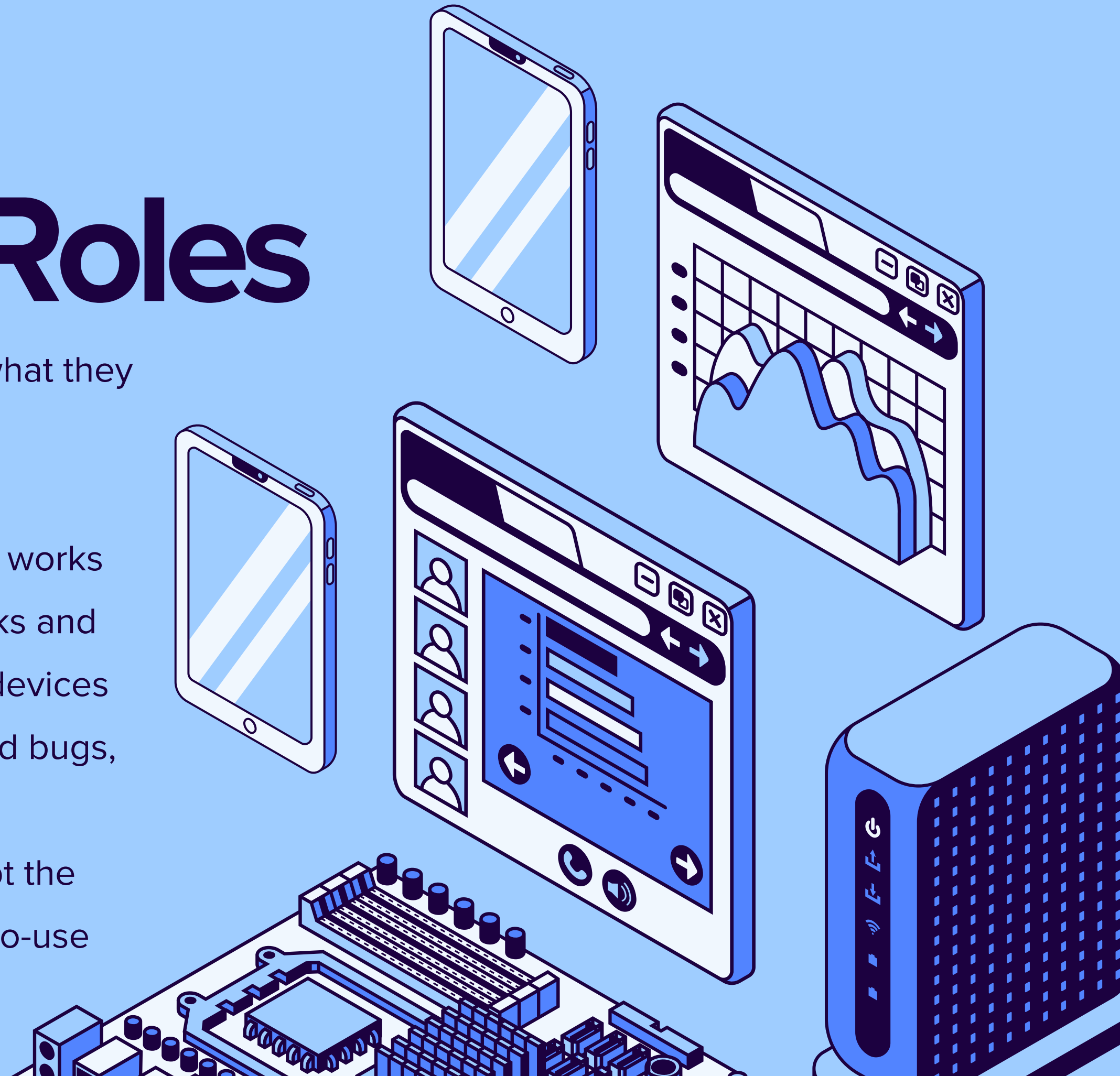
AC, CTI EN, 3<sup>RD</sup> YEAR

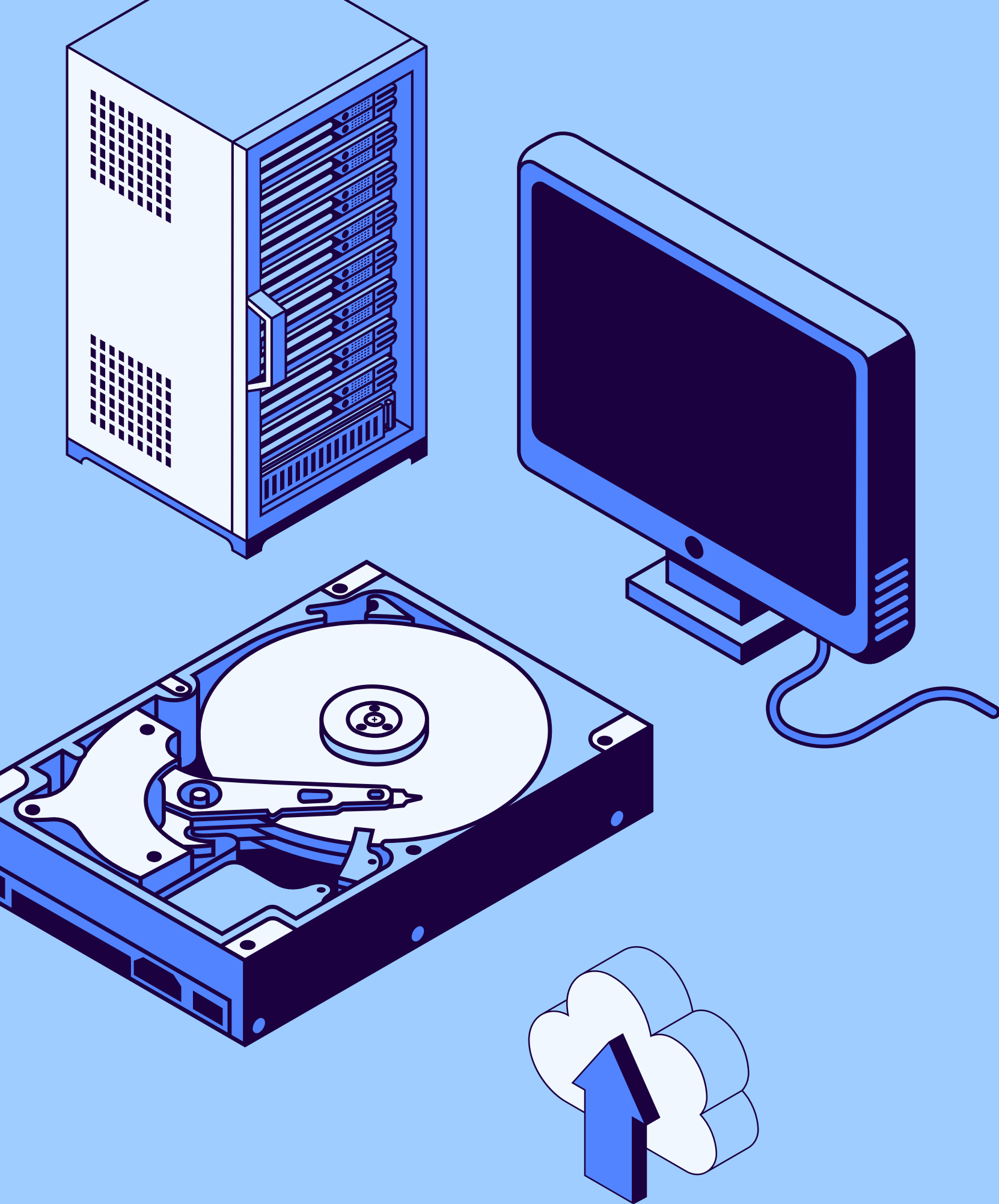


# Our Team & Roles

To keep things organized, everyone focused on what they do best:

- Rares – Software Developer - Built the C/C++ application and made sure everything actually works
- Darius – Hardware - Tested different USB sticks and checked how the program behaves with real devices
- Dragos – Tester - Tried to break the app, found bugs, and helped during the live demo
- Ionut – Project Manager & Interface / UX - Kept the team organized and designed a simple, easy-to-use interface





# Our Project

The USB Information Tool is a small console program made in C/C++ that checks a USB flash drive when it's plugged into a Windows computer.

It shows useful info like:

- USB device name
- Total storage size
- Free space available
- Basic device details

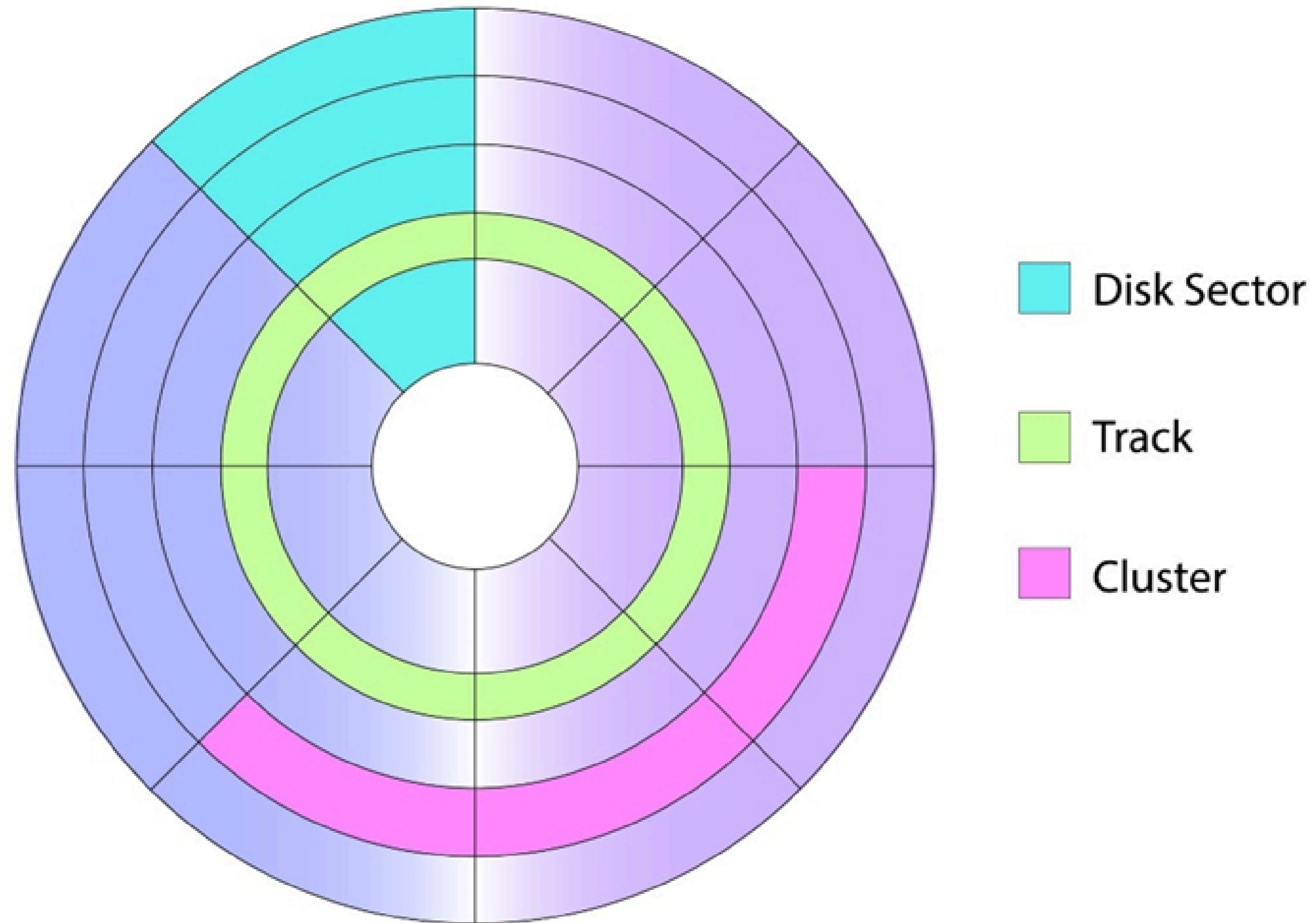
We made this project as a team assignment to learn how USB devices work and how a program can read information from external hardware.

# USB Storage Fundamentals

- Sector: Smallest physical unit that can be read or written (512 bytes standard).
- Sectors are addressed sequentially using Logical Block Addressing (LBA).
- Cluster: A logical grouping of sectors used by the FAT32 file system.
- Larger clusters improve speed but increase wasted space (slack space).
- Even a 1 KB file occupies a full cluster, affecting storage efficiency.



# DISK DRIVE SECTORS



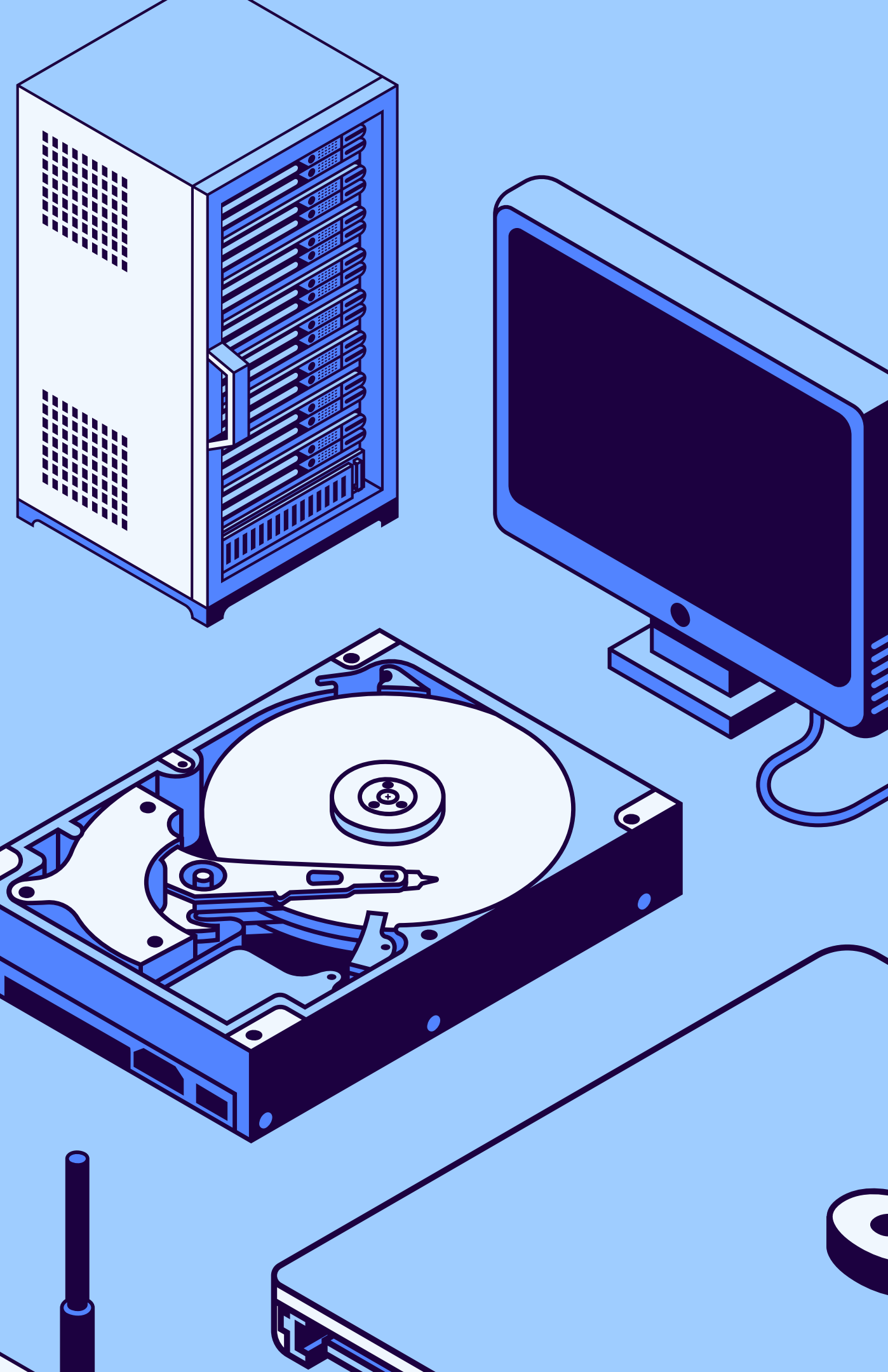


- Boot Sector (Sector 0) initializes the entire file system structure.
- The BIOS Parameter Block (BPB) defines how the USB must be interpreted.
- FAT Region functions as a linked list mapping file cluster chains.
- Multiple FAT copies exist for redundancy and recovery.
- The Data Region stores both directory entries and actual file data.

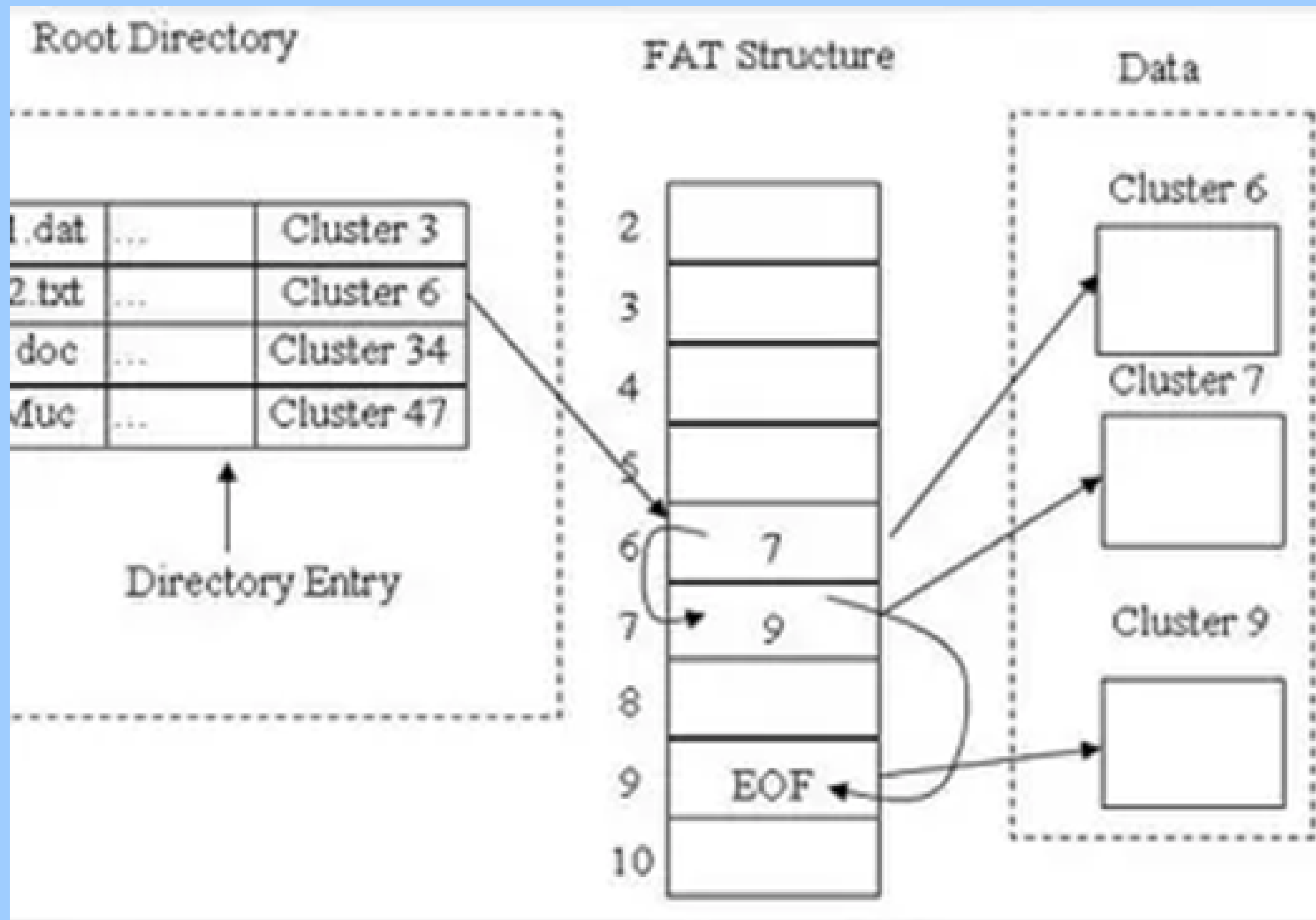
# FAT32 File System Architecture

# FAT32 Directory Entries Explained

- Each directory entry is exactly 32 bytes for hardware alignment.
- Entries store filename, extension, attributes, timestamps, and size.
- The Attribute byte is a bitmask defining file behavior and visibility.
- Long File Names (0x0F) use multiple chained entries and are skipped.
- Dates and times are compressed to save space and decoded via bit shifts.



# Structural diagram of a standard USB formatted with FAT32



# Safe USB Cleaning Mechanism

- File deletion does not erase data, only metadata references.
- The filename's first byte is changed to 0xE5 to mark deletion.
- FAT entries for the file's clusters are marked as free (0x00000000).
- Actual data remains until overwritten, enabling file recovery.
- Incorrect deletion can break FAT chains and corrupt the file system.





# Rules of Engagement for Safe Cleaning

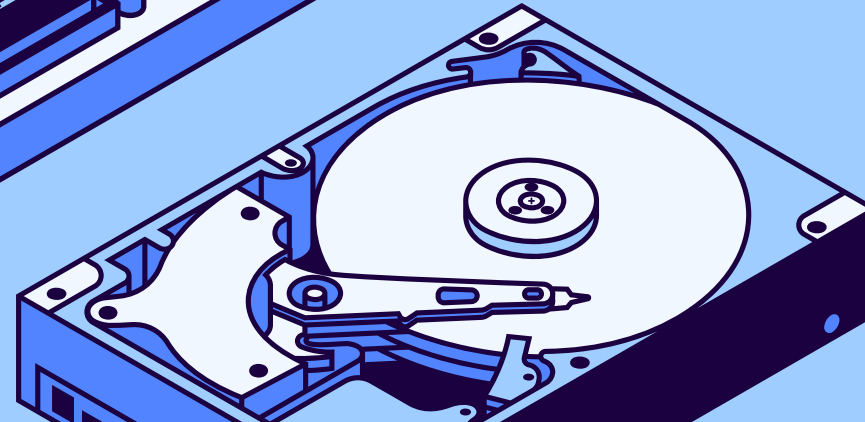
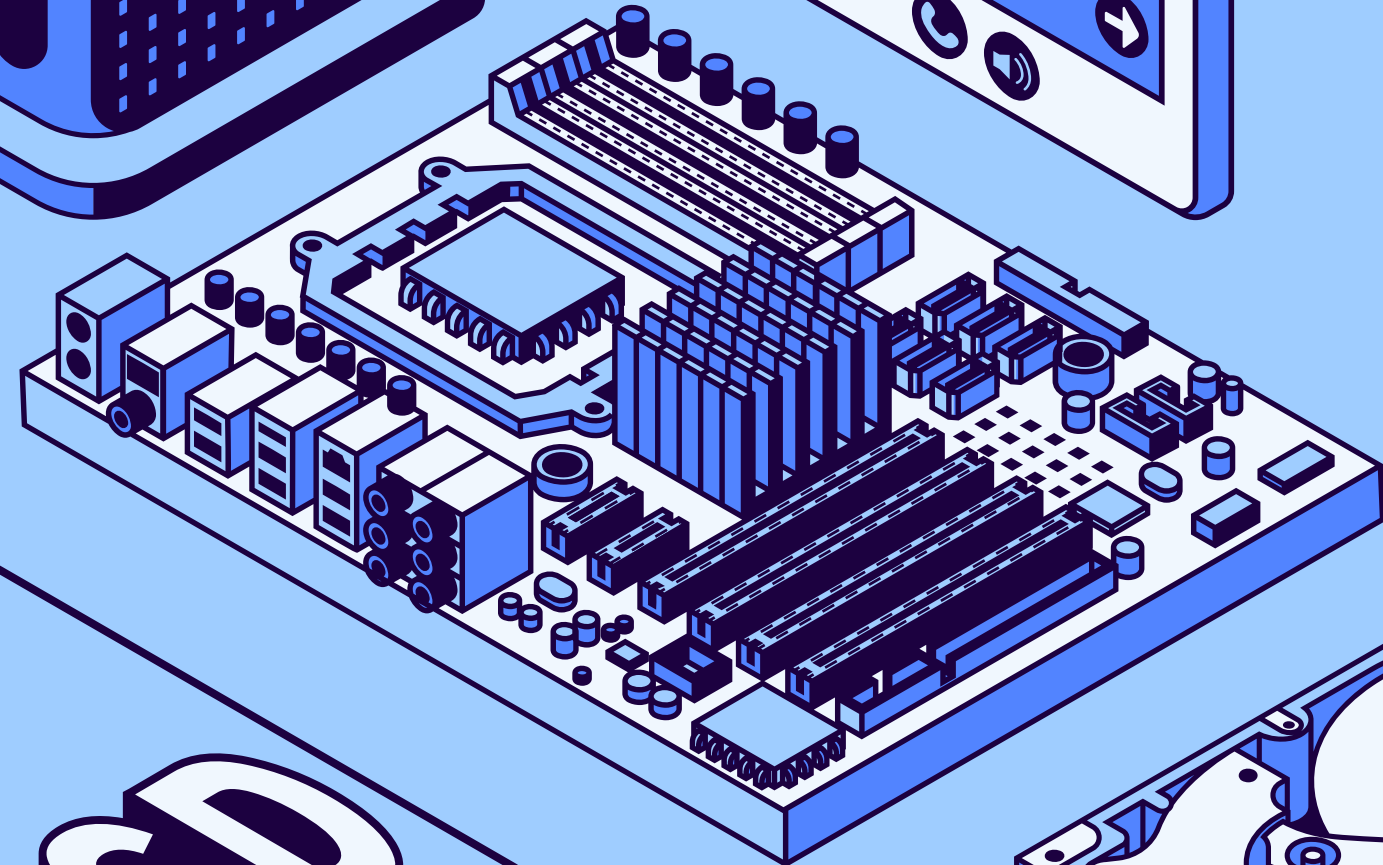
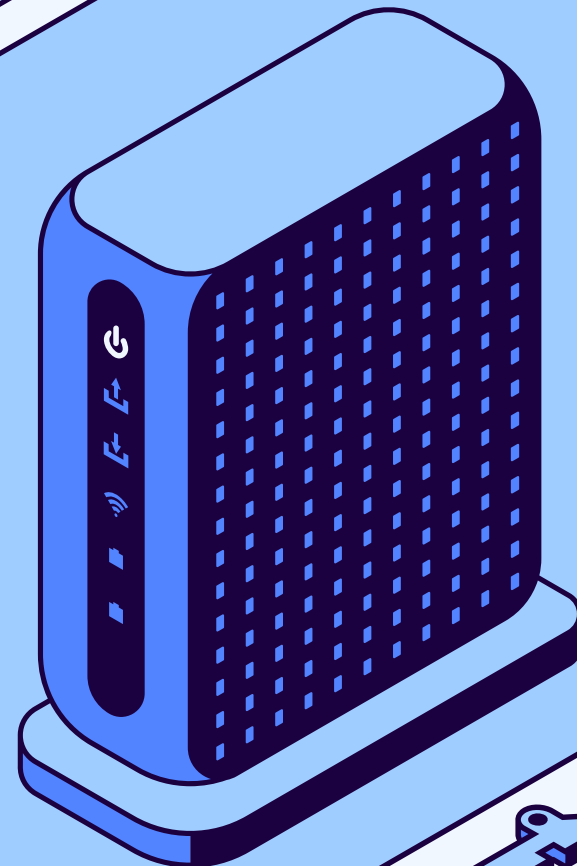
- System and Hidden files must never be modified or deleted.
- Root directory entries such as '.' and '..' must always be preserved.
- Only known temporary file extensions are eligible for deletion.
- Empty directories may be removed only after validation.
- Strict filtering prevents OS errors and USB corruption.

# Resolving technical problems

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

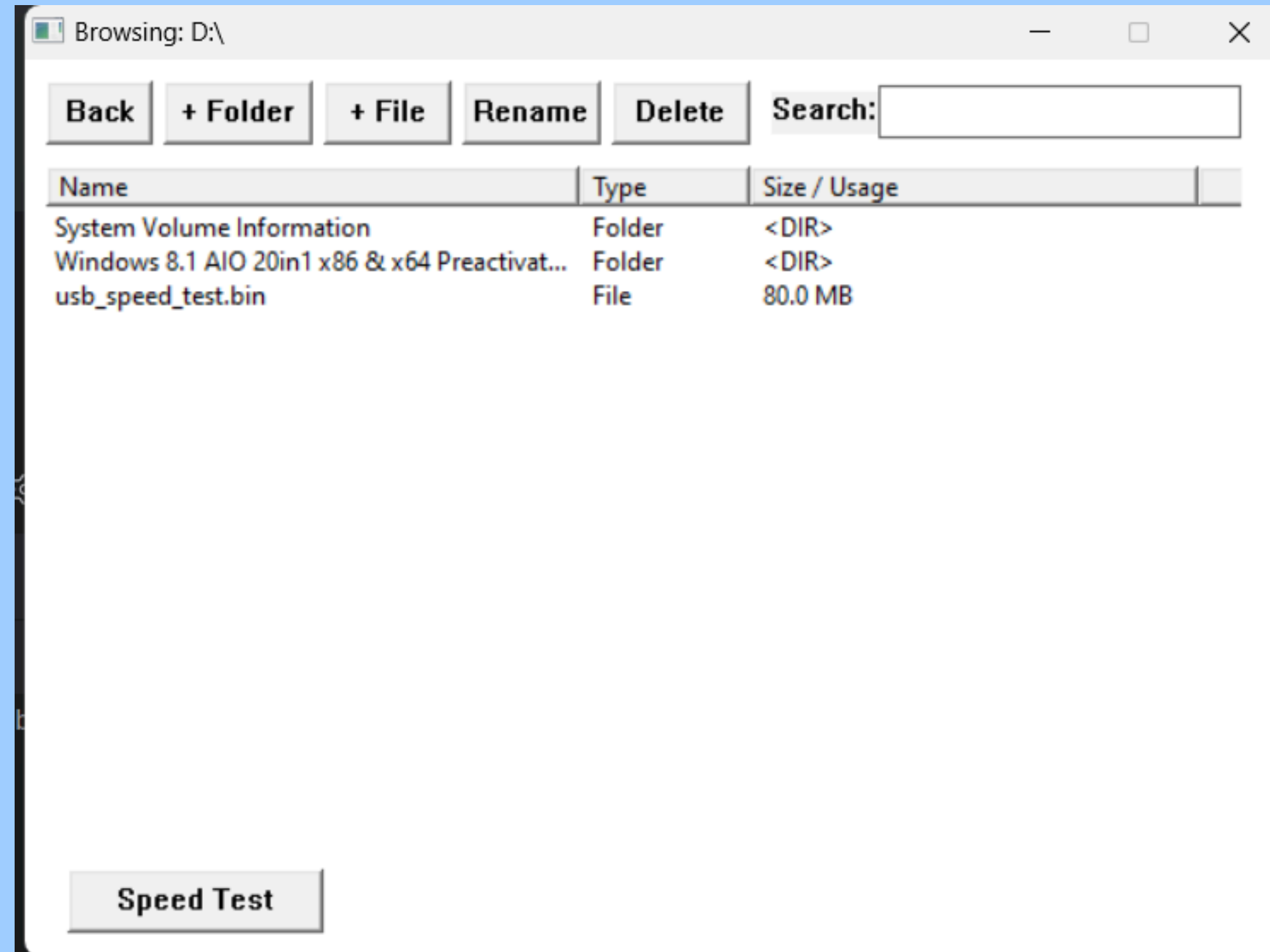
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.





Now let's talk  
about our  
application...

# Here is the first look



The UI - Developed by Ionuț and Rareș  
Speed Test and the functionality of the program - Dragoș



### **WINDOWS API (WIN32)**

- WINDOWS.H – CORE OS, WINDOW MANAGEMENT, MESSAGES
- USER32 – GUI WINDOWS, BUTTONS, MESSAGE LOOP
- GDI32 – INTERNAL GRAPHICS SUPPORT

### **COMMON CONTROLS**

- COMMCTRL.H
- COMCTL32.LIB

### **USED FOR LISTVIEW (FILE & USB DISPLAY)**

- SHELL & SYSTEM
- SHELLEXECUTE (SHELL32) – OPEN FILES
- KERNEL32 – THREADING, SYSTEM CALLS

# **Libraries and APIs used**

# Testing and Validation

## Testing & Validation

- USB Detection Testing
  - Plug/unplug USB devices → automatic refresh (WM\_DEVICECHANGE)
  - Verified correct drive detection and labeling
- File System Operations Testing
  - Create / rename / delete files and folders
  - Manual validation through ListView updates
- Performance Testing
  - USB read/write speed test (background thread)
  - UI remains responsive during testing
- Error & Stability Testing
  - Invalid actions handled with warnings
  - Logging used to track errors and events





**Now time  
for a live  
demo...**

# Thank You

