

Index

Sr. No.	Practical name	Page No.
1	Message encryption over a channel	2
2	Custom logging module	7
3	Searching files in a given directory	10
4	Searching a word in a given file	13
5	Virus that eats disk space	16
6	Disk backup	19
7	Forensic images of a disk	28
8	Retrieving deleted files	35
9	Registry Editor	43

Practical 1

Aim: Create a Java file to send an encrypted message from the sender end and decrypt it at the receiver's end.

Source Code:

Sender.java:

```
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.Random;
import java.util.Scanner;

/**
 * Sender: Sends an encrypted message and generated key
 * to the receiver.
 * Uses Sockets for communication.
 */
public class Sender {
    /**
     * @param args Command line arguments
     */
    public static void main(String[] args) {
        int counter = 0;
        String cipherText = "", key = "";
        Random random = new Random();
        Scanner scanner = new Scanner(System.in);

        try {
            Socket socket = new Socket("localhost", 6017);
```

```

DataOutputStream dataOutputStream = new
DataOutputStream(socket.getOutputStream());
System.out.println("Enter message: ");
String message = scanner.nextLine();

/*
 * Code for encryption.
 * Working:
 * 1. Generate an array of n (length of the message) random
numbers.
 * 2. Add the codePoints of the message with the array
sequentially.
 * 3. Append the typecasted character to the ciphertext.
 */
int[] keyArray = new int[message.length()];
for (char messagePart : message.toCharArray()) {
keyArray[counter] = random.nextInt(50);
key += Integer.valueOf(keyArray[counter]) + ":";
cipherText += (char)(messagePart + keyArray[counter]);
counter++;
}

System.out.println("Message: " + message);
System.out.println("Generated key: " + key);
System.out.println("Encrypted message: " + cipherText);

dataOutputStream.writeUTF(cipherText);
dataOutputStream.writeUTF(key);

scanner.close();
dataOutputStream.flush();
dataOutputStream.close();
socket.close();
}

```

```

catch (UnknownHostException e) {
    System.err.println("Error: Host not found.");
    e.printStackTrace();
}
catch (IOException e) {
    System.err.println("IOError: Some I/O operations could not
be performed.");
    e.printStackTrace();
}
}
}
}

```

Receiver.java:

```

import java.io.DataInputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

/**
 * Receiver: Receives an encrypted message and key from the
 * sender
 * and decrypts it.
 * Uses Sockets for communication.
 */
public class Receiver {
    public static void main(String[] args) {
        String message = "";
        int counter = 0;

        try {
            ServerSocket serverSocket = new ServerSocket(6017);
            Socket socket = serverSocket.accept();
            DataInputStream dataInputStream = new

```

```

DataInputStream(socket.getInputStream());

String cipherText = dataInputStream.readUTF();
String key = dataInputStream.readUTF();

/*
 * Code for decryption.
 * Working:
 * 1. Split the key string using the ':' delimiter and
    convert it into an integer.
 * 2. Subtract the array values from the codePoints
    sequentially.
 * 3. Append the typecasted character to the message.
 */
int[] keyArray = new int[cipherText.length()];
for (String keyPart : key.split(":")) {
    keyArray[counter] = Integer.parseInt(keyPart);
    message += (char)(cipherText.charAt(counter) -
    keyArray[counter]);
    counter++;
}

System.out.println("Ciphertext: " + cipherText);
System.out.println("Key: " + key);
System.out.println("Message: " + message);

dataInputStream.close();
socket.close();
serverSocket.close();
}
catch (IOException e) {
    System.err.println("IOError: Some I/O operations could not
    be performed");
    e.printStackTrace();
}

```

```
}  
}  
}
```

Output:

Sender

```
>java Sender  
Enter message:  
Hello, World!  
Message: Hello, World!  
Generated key: 41:26:24:48:15:18:37:16:10:27:13:22:31:  
Encrypted message: q[]??~>Egy?yz@
```

Receiver

```
>java Receiver  
Ciphertext: q[]??~>Egy?yz@  
Key: 41:26:24:48:15:18:37:16:10:27:13:22:31:  
Message: Hello, World!
```

Practical 2

Aim: Create a Java file to create a logger.

Source Code:

```
import java.io.FileWriter;
import java.io.IOException;
import java.time.LocalDateTime;
import java.util.Random;
import java.time.format.DateTimeFormatter;
public class CustomLogger {
    FileWriter fileWriter;

    public CustomLogger(String filePath, boolean appendMode) {
        try {
            fileWriter = new FileWriter(filePath, appendMode);
        }
        catch (IOException e) {
            System.err.println("IOError: File could not be opened");
            e.printStackTrace();
        }
    }

    public void writeLog(String message, String intensity) {
        String datetime = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss").format(LocalDateTime.now());
        try {
            fileWriter.write(datetime + "\t\t" + message + "\t\t" +
                intensity + "\n");
            fileWriter.flush();
        }
        catch (IOException e) {
```

```

System.err.println("IOException: Log could not be written");
e.printStackTrace();
}
}

public void close() {
try {
fileWriter.close();
} catch (IOException e) {
System.err.println("IOException: File could not be closed");
e.printStackTrace();
}
}

public static void main(String[] args) {
CustomLogger customLogger = new CustomLogger("log.txt",
true);
String[] intensity = {"INFO", "WARNING", "ERROR",
"CRITICAL"};
Random random = new Random();

for (int i = 0; i < 10; i++) {
customLogger.writeLog("Log " + i,
intensity[random.nextInt(4)]);
}

customLogger.close();
}
}

```

Output:

1	2022/07/06 21:07:04	Log 0	INFO
2	2022/07/06 21:07:05	Log 1	INFO
3	2022/07/06 21:07:05	Log 2	ERROR
4	2022/07/06 21:07:05	Log 3	CRITICAL
5	2022/07/06 21:07:05	Log 4	INFO
6	2022/07/06 21:07:05	Log 5	ERROR
7	2022/07/06 21:07:05	Log 6	CRITICAL
8	2022/07/06 21:07:05	Log 7	ERROR
9	2022/07/06 21:07:05	Log 8	ERROR
10	2022/07/06 21:07:05	Log 9	WARNING
11			

Practical 3

Aim: Create a Java file to search for files in a given directory.

Source Code:

```
import java.io.File;
import java.util.Scanner;

public class DirectorySearcher {
    private String directoryPath;

    /**
     * @param directoryPath Absolute path of the directory
     * Creates a directorySearcher object with a specified
     * directory path.
     */
    public DirectorySearcher(String directoryPath) {
        this.directoryPath = directoryPath;
    }

    /**
     * @param filter Filter to be applied
     * Searches the directory for filenames starting with given
     * filter.Ignores subdirectories.
     */
    public void search(String filter) {
        File file = new File(directoryPath);
        File[] fileArray = file.listFiles();

        for (File file2 : fileArray) {
            if (file2.isDirectory()) {
                continue;
            }
        }
    }
}
```

```

}
if (file2.getName().startsWith(filter)) {
    System.out.println(file2.getName());
}
}
}

/**
 * @param args Command line arguments
 * Driver code.
 */
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.println("Enter a directory > ");
    String directoryPath = scanner.nextLine();

    DirectorySearcher directorySearcher = new
    DirectorySearcher(directoryPath);

    System.out.println("Enter filter > ");
    String filter = scanner.nextLine();

    directorySearcher.search(filter);

    scanner.close();
}
}

```

Output:

```
>java DirectorySearcher
Enter a directory >
/Documents/Practicals/Temp
Enter filter >
D
DirectorySearcher.class
DirectorySearcher.java
```

Practical 4

Aim: Create a Java file to search for files in a given directory.

Source Code:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class FileSearcher {
    private String absFileName;

    public FileSearcher(String absFileName) {
        this.absFileName = absFileName;
    }

    public boolean search(String word) {
        boolean found = false;

        try {
            File file = new File(absFileName);
            Scanner scanner = new Scanner(file);

            while (scanner.hasNext()) {
                if(scanner.nextLine().indexOf(word) != -1) {
                    found = true;
                }
            }

            scanner.close();
        } catch (FileNotFoundException e) {
            System.out.println("File not found.");
        }
    }
}
```

```

e.printStackTrace();
}

return found;
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.println("Enter a file name > ");
    String fileName = scanner.nextLine();

    FileSearcher fileSearcher = new FileSearcher(fileName);

    System.out.println("Enter a word filter > ");
    String word = scanner.nextLine();
    scanner.close();

    boolean found = fileSearcher.search(word);
    if (found) {
        System.out.println("Word found");
    } else {
        System.out.println("Word not found");
    }
}
}
}

```

Output:

```
>java FileSearcher
Enter a file name >
log.txt
Enter a word filter >
Log
Word found
```

```
>java FileSearcher
Enter a file name >
log.txt
Enter a word filter >
Not
Word not found
```

Practical 5

Aim: Create a Java file to create a virus that eats disk space.

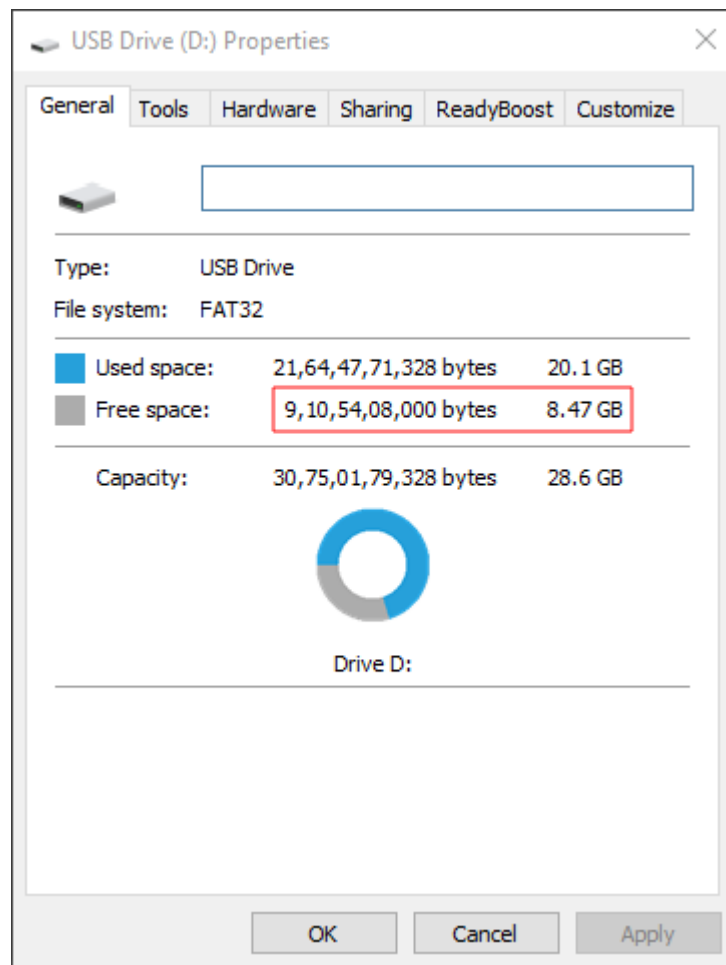
Source Code:

```
import java.io.FileWriter;
import java.io.IOException;

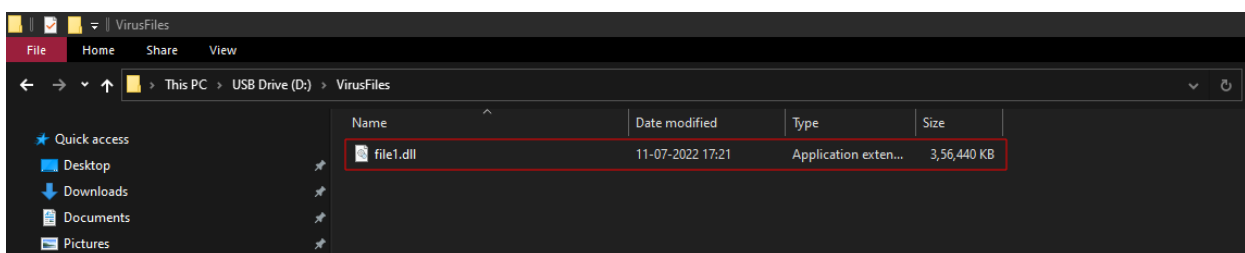
public class VirusExample {
    /**
     * @param args Command-line arguments.
     * @throws IOException if file cannot be opened.
     *
     * Creates a file named file1.dll in append mode and
     * repeatedly
     * appends "Virus" into it.
     */
    public static void main(String[] args) throws
    IOException {
        FileWriter fileWriter = new
        FileWriter("D:/VirusFiles/file1.dll", true);
        while (true) {
            fileWriter.write("Virus");
        }
    }
}
```

Output:

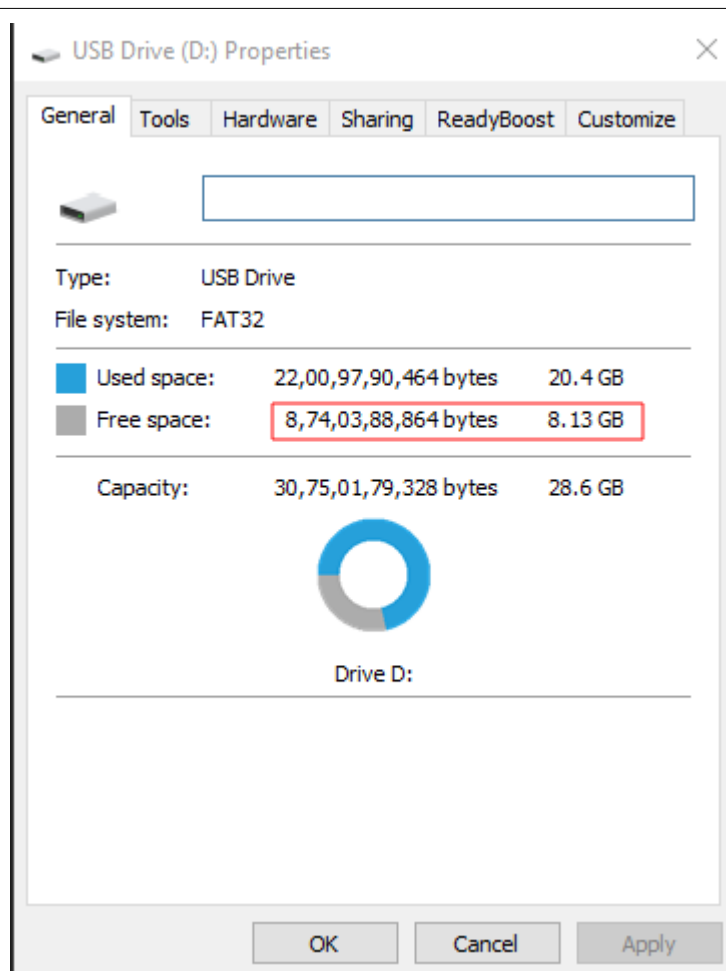
- Before:



- Generated file:



- After:

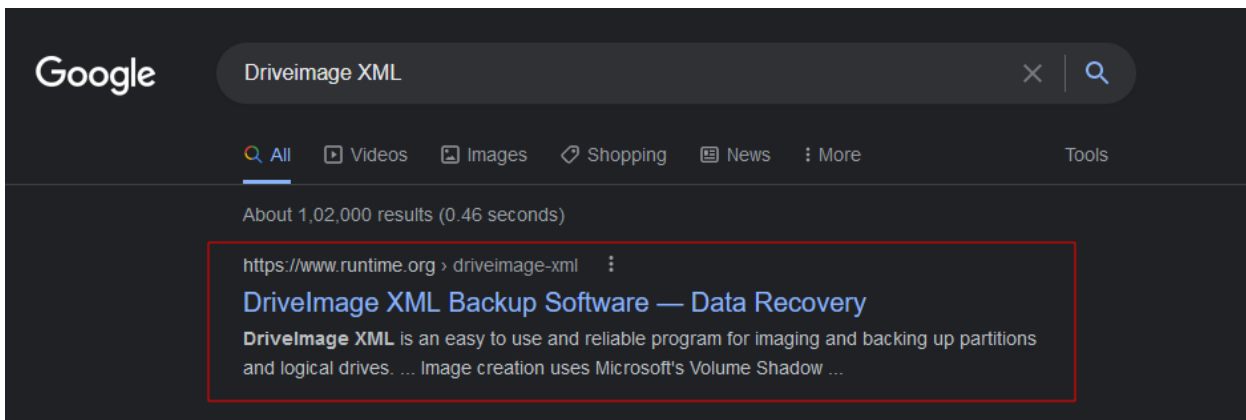


Practical 6

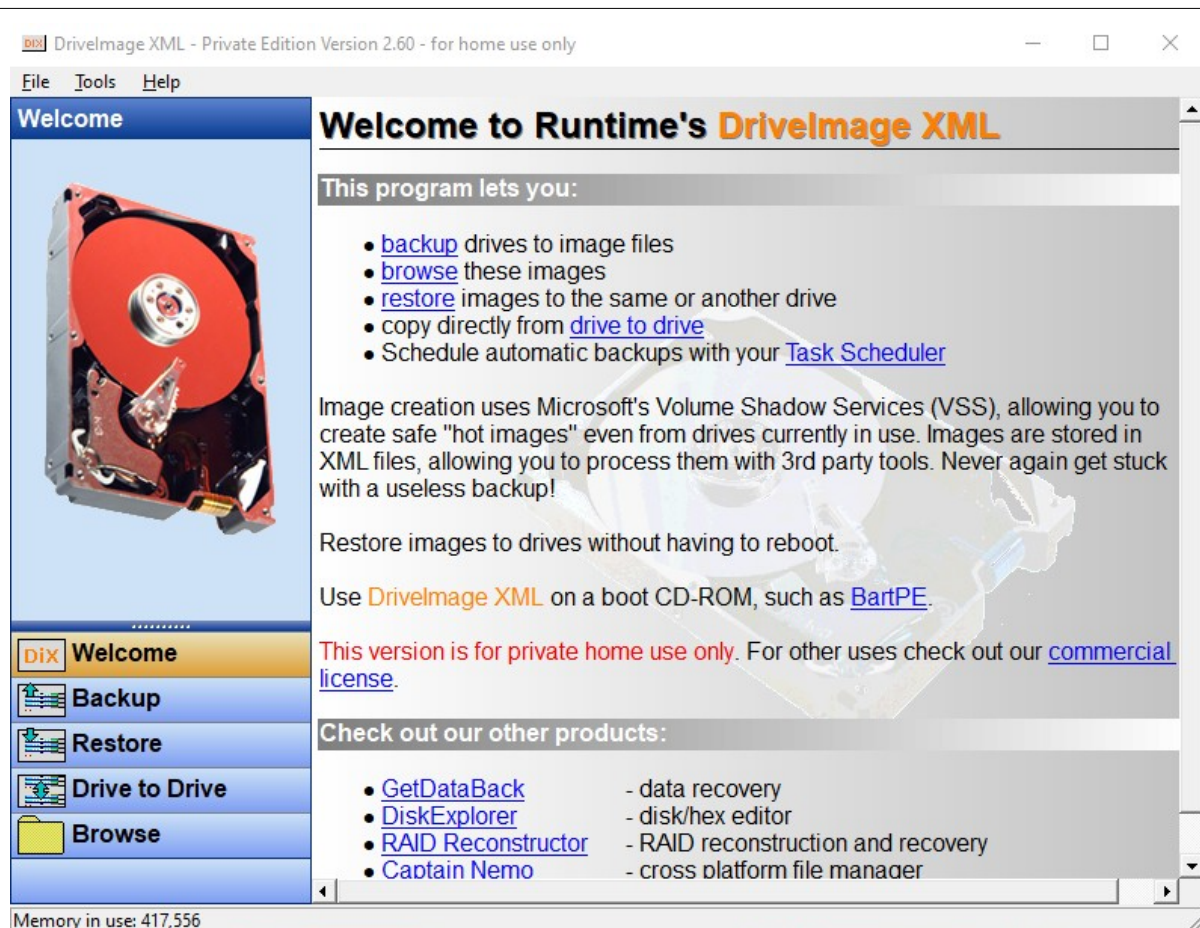
Aim: Create a backup of a disk using DriveImage XML.

Procedure:

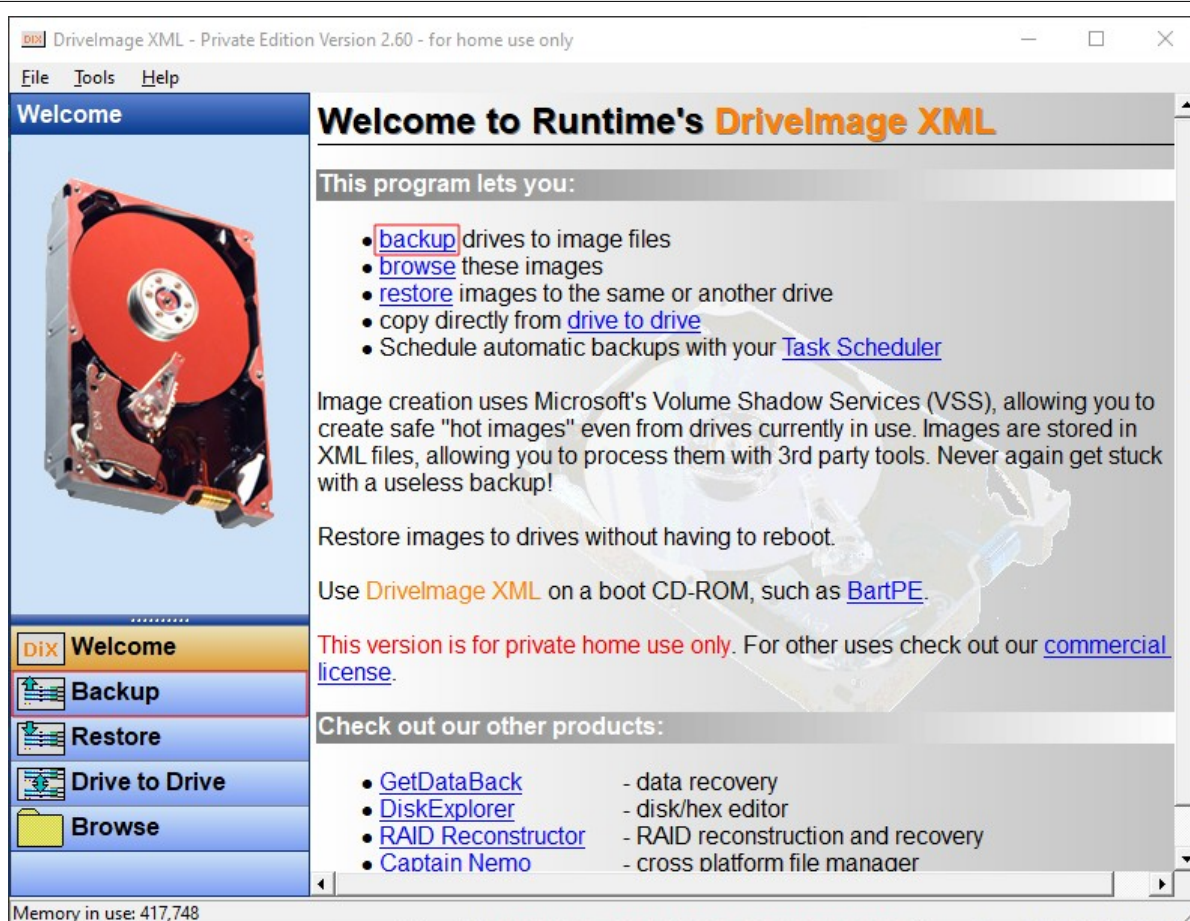
- Download and install **DriveImage XML** from this [link](#). A quick web search should lead you to this website:



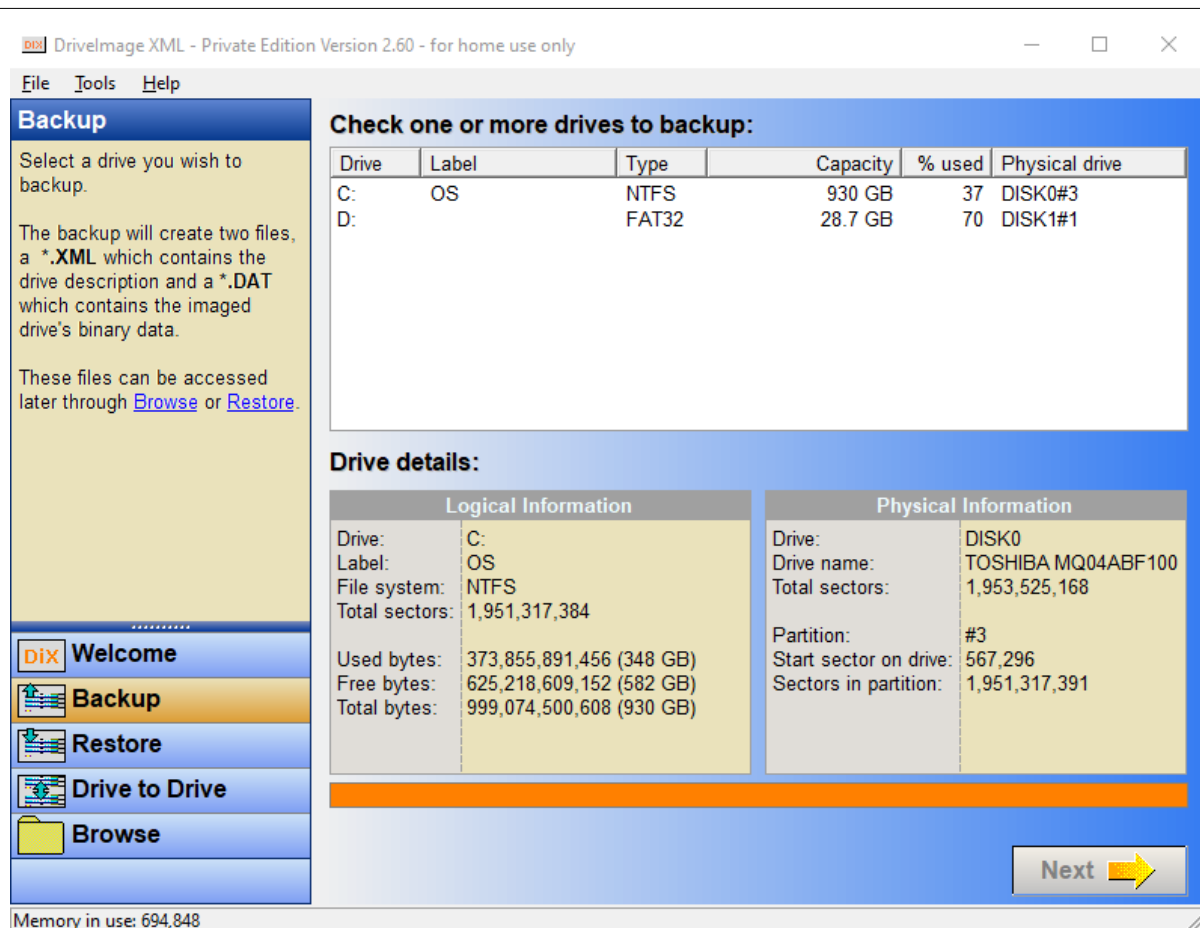
- After opening **DriveImage XML**, you will be presented with this screen:



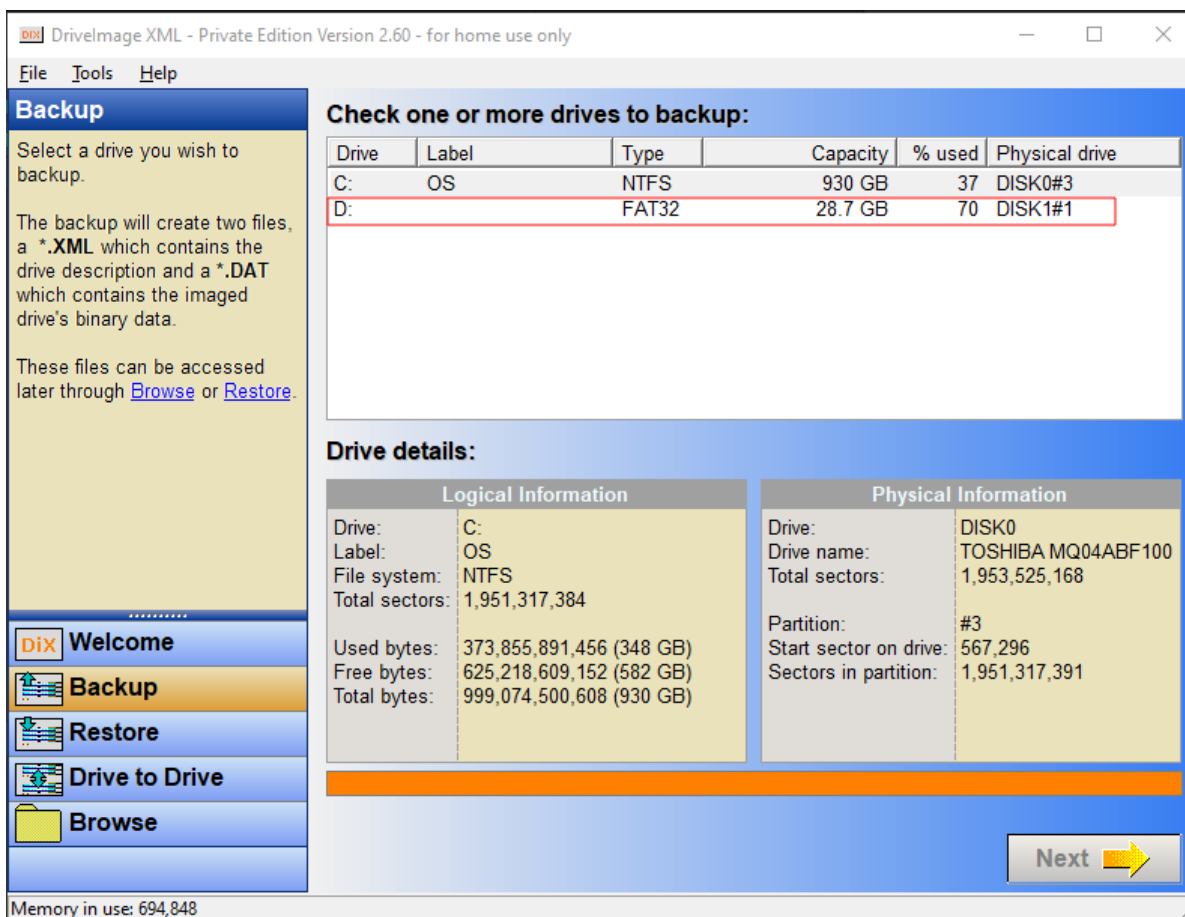
- You can either use the **Backup** hyperlink or the **Backup** button to start the backup operation:

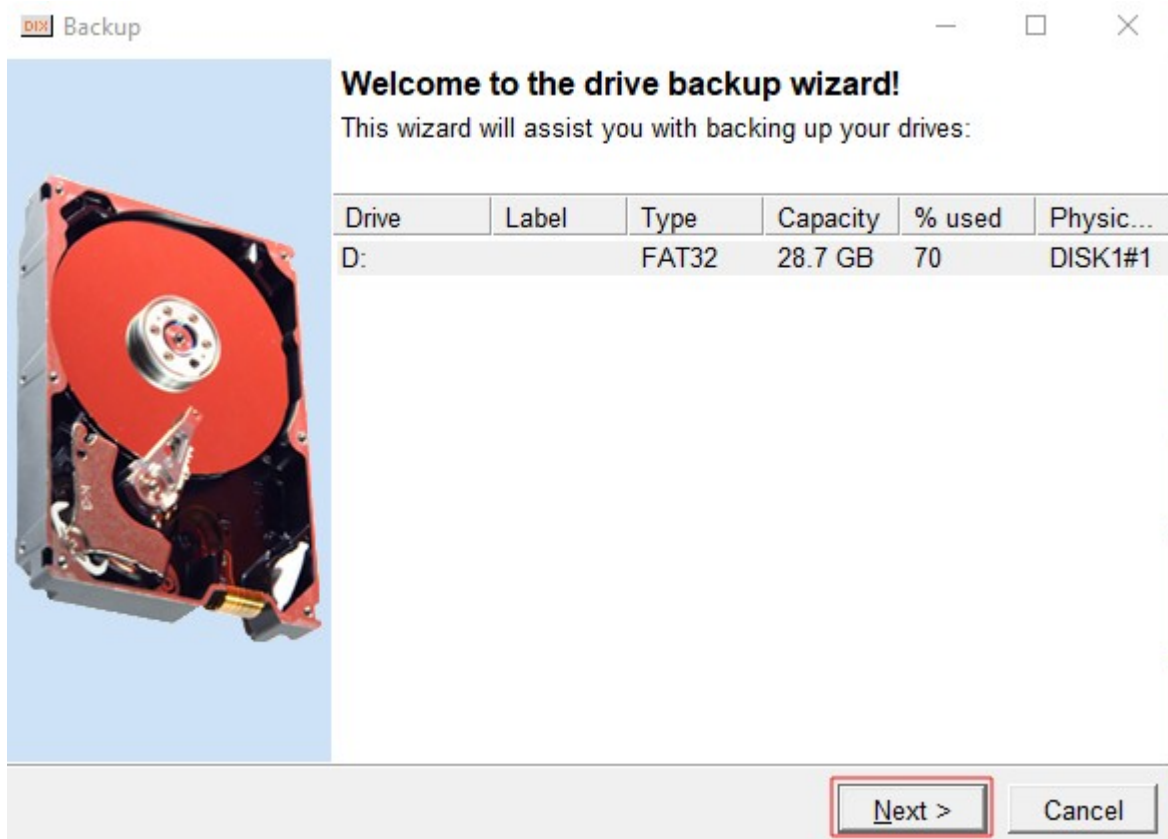


- After clicking on either of the two options listed above, it should show you a list of all the disk(s) present on your system:



- Choose one (or multiple) disk(s) to image. In this exercise, Disk D is chosen for creating a backup. After clicking on "Next", the Backup wizard will be displayed. After confirming your selection, click on [Next](#):





- Confirm other details such as Output location and other settings and when comfortable, click on [Next](#).

Backup
Select a backup location and imaging options.

Directory:

Files:

Drive	File name
D:	Drive_D

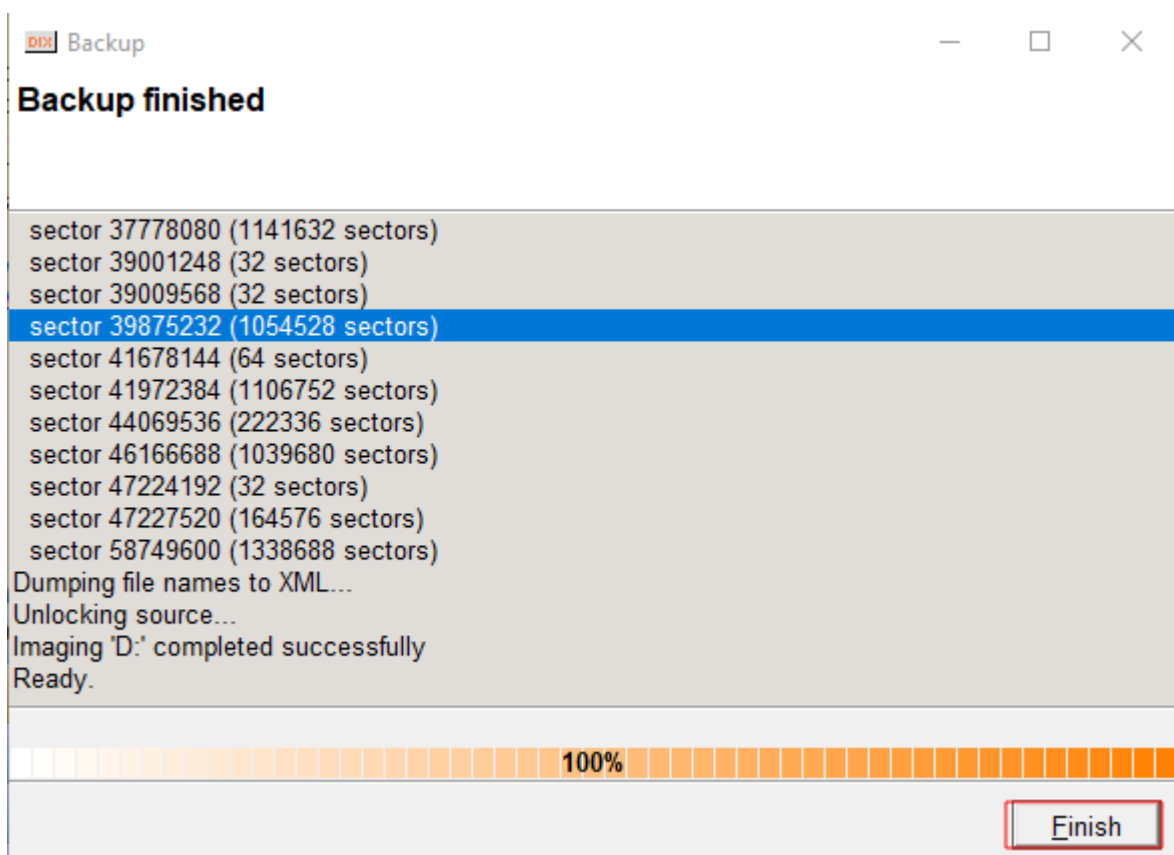
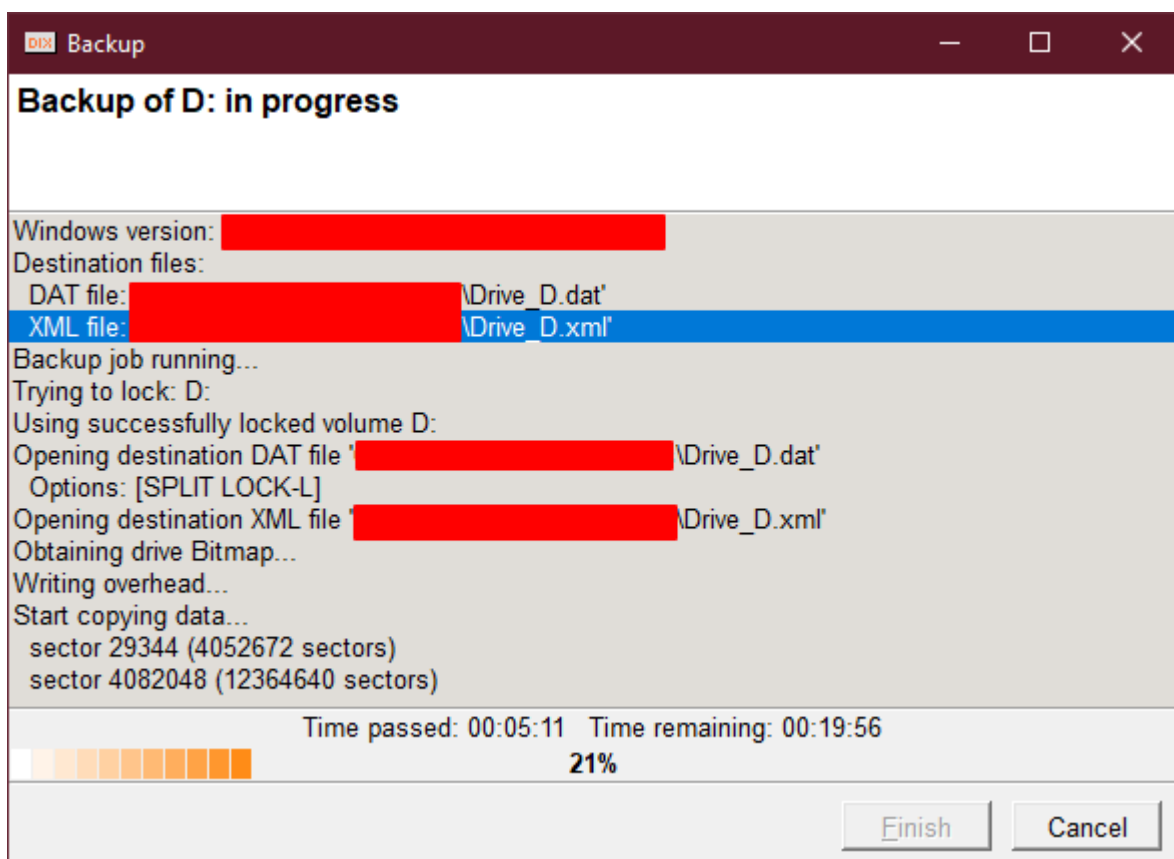
Options:

☐ Raw mode
☒ Split large files
Compression:

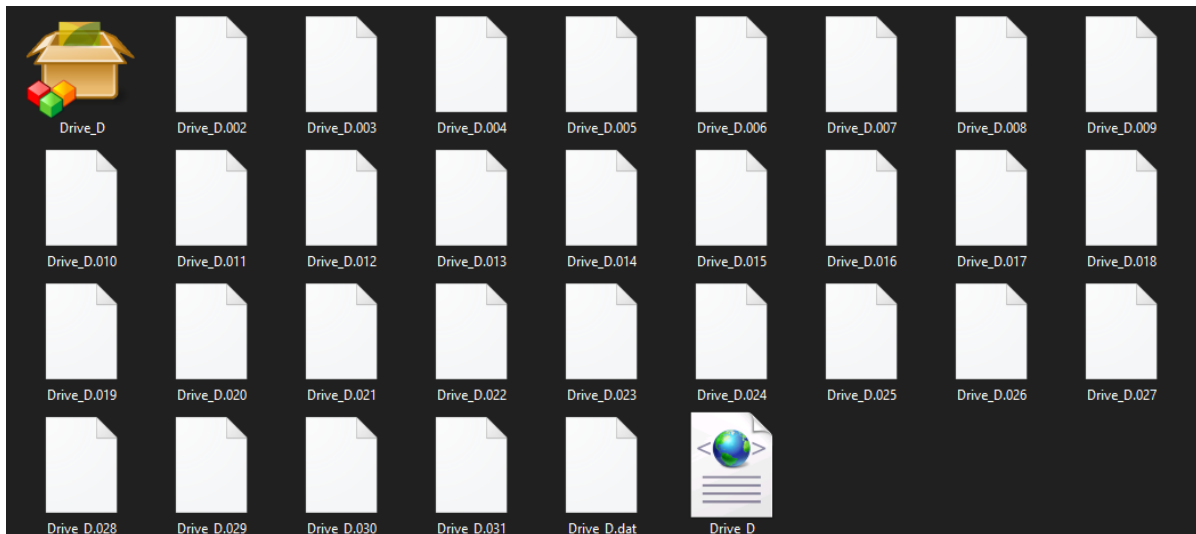
Hot Imaging Strategy:
☒ Try Volume Locking first
☐ Try Volume Shadow Services first

< Back Next > Cancel

- The backup process will start shortly. Wait until the progress bar reaches 100%. After which click on [Finish](#).



- The following files will be generated in the destination folder.



- The generated XML file has the following text:

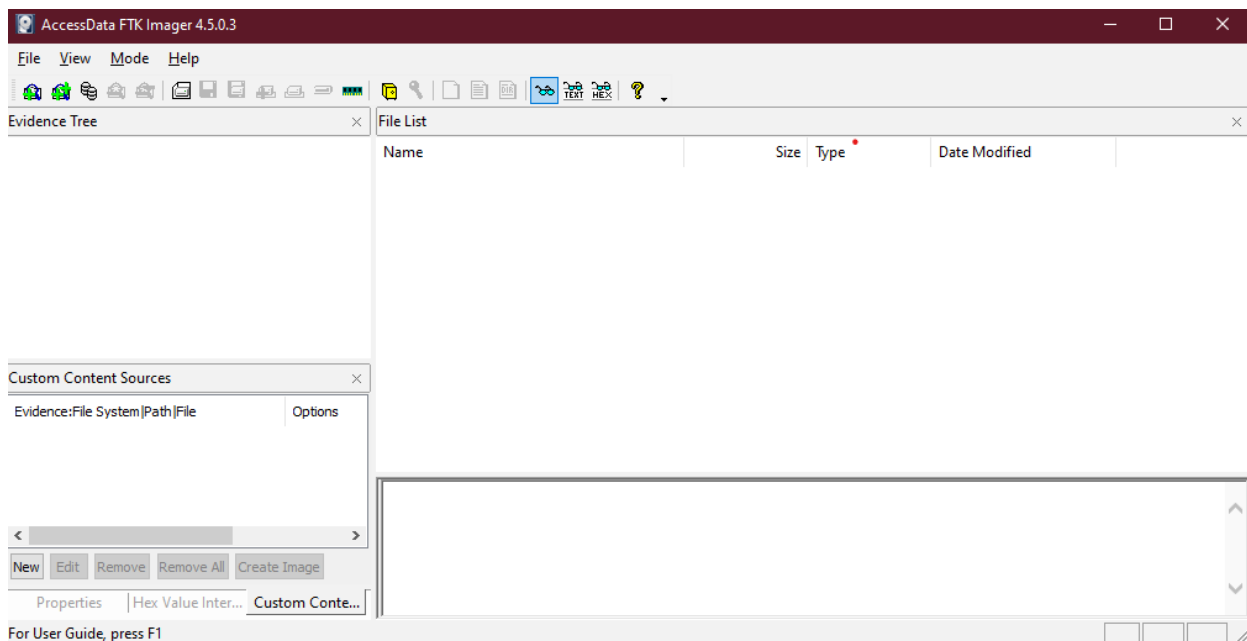
```
<?xml version="1.0" encoding="UTF-8"?>
<driveimage creator="DriveImage XML - Private Edition" version="Version 2.60" time="2022-07-11T18:05:00" id="1">
  <!--
This XML document describes a drive image created with Runtime Software's DriveImageXML.
It uses the following XML tags:
<driveimage> - the root node
  Attributes:
    creator - application that created this image (usually "DriveImage XML")
    version - version of the application that created this image (e.g. "Version 1.00")
    time - date and time this image was created (e.g. "2005-09-08T23:40:03.767-08:00")
    destpath - path where this image was originally written to (e.g. "X:\backup\")
    filename - original name of this image file (e.g. "Drive_C")
    compressed - accompanying binary file is compressed
    raw - the image is a raw image
    split - accompanying binary file is split in CD-ROM sized files
    password - a password will be required for browsing or restoring of the image
    id - a unique identifier for this image
  <drive> - opening tag for the drive that follows
  <driveletter> - the original drive letter of the imaged drive
  <drivelabel> - the label of the imaged drive
  <totalspace> - capacity of the imaged drive in bytes
  <freespace> - unused space on the imaged drive in bytes
  -->
  <driveletter>C
  <drivelabel>C:
  <totalspace>1073741824
  <freespace>1073741824
  <imagefile>Drive_D.dat
  <imagefilecompressed>Drive_D.002
  <imagefilesplit>Drive_D.003
  <imagefilepassword>
  <imagefileid>1
  </drive>
</driveimage>
```

Practical 7

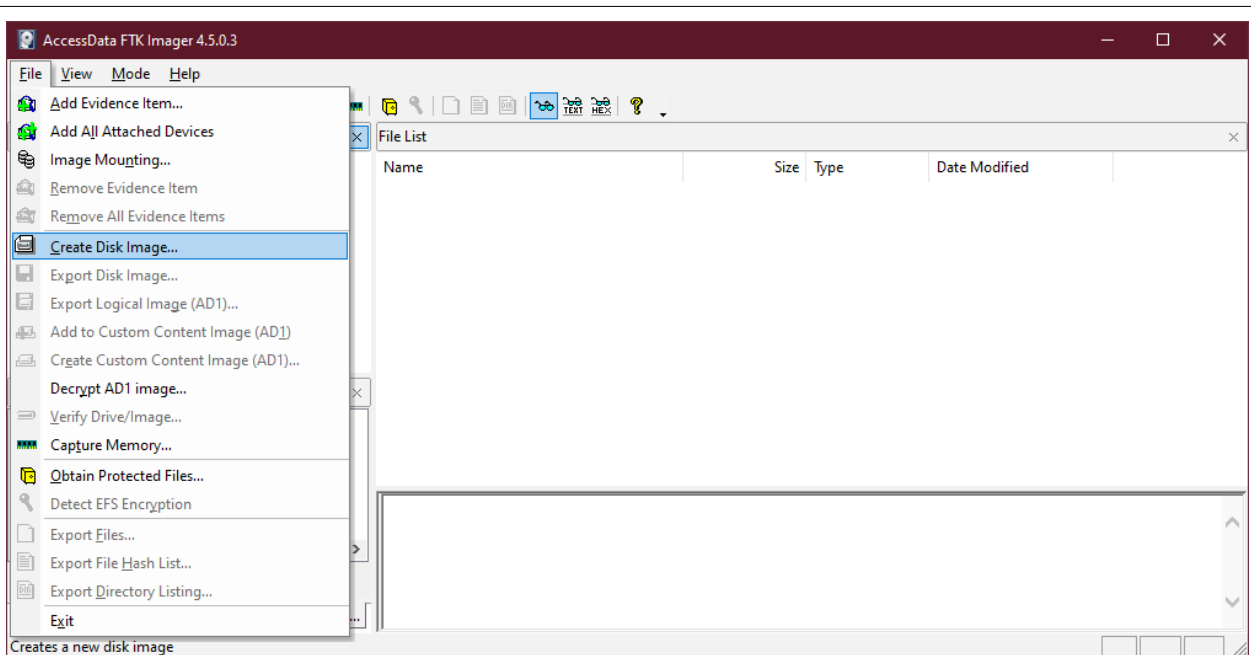
Aim: Create a forensic image of a digital device from volatile data such as memory.

Procedure:

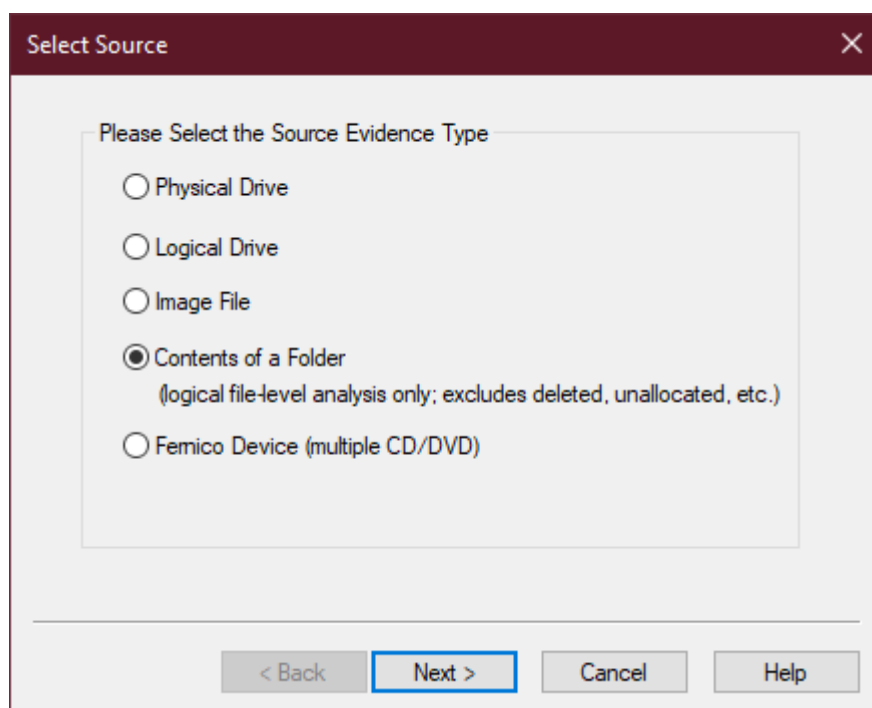
- Download and install **AccessData® FTK® Imager** from this [link](#). Launching the application will display a screen similar to this:



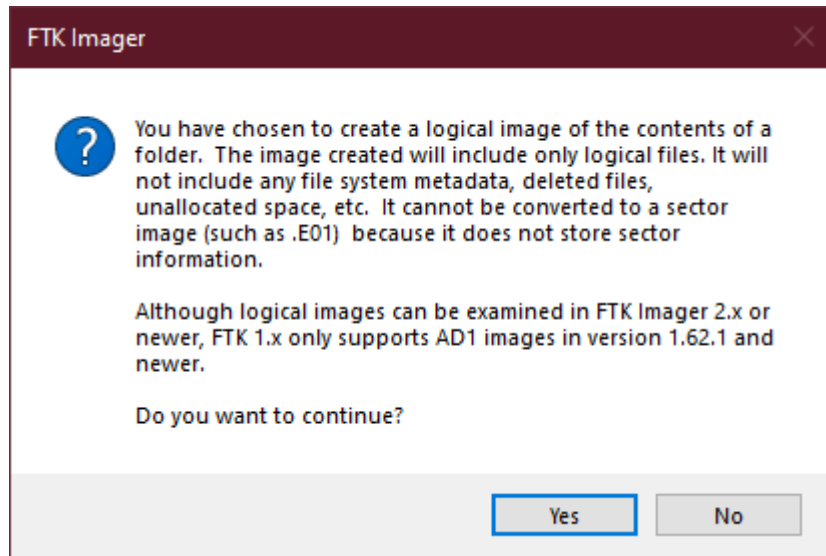
- Now, navigate to **File > Create Disk Image....**



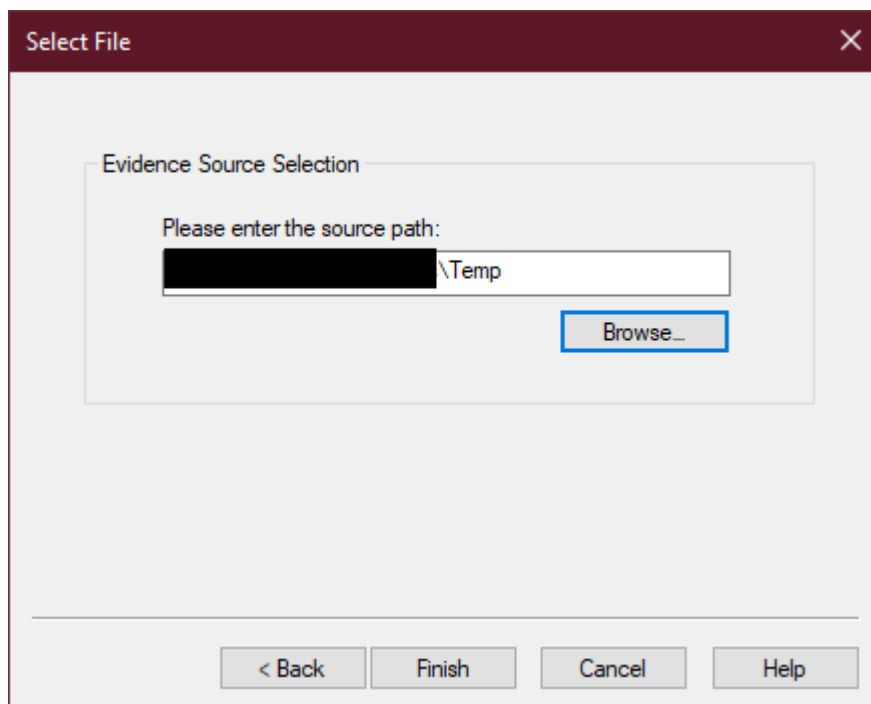
- This should bring up a new window. Select the **Contents of a Folder** option for the source. Click on **Next**.



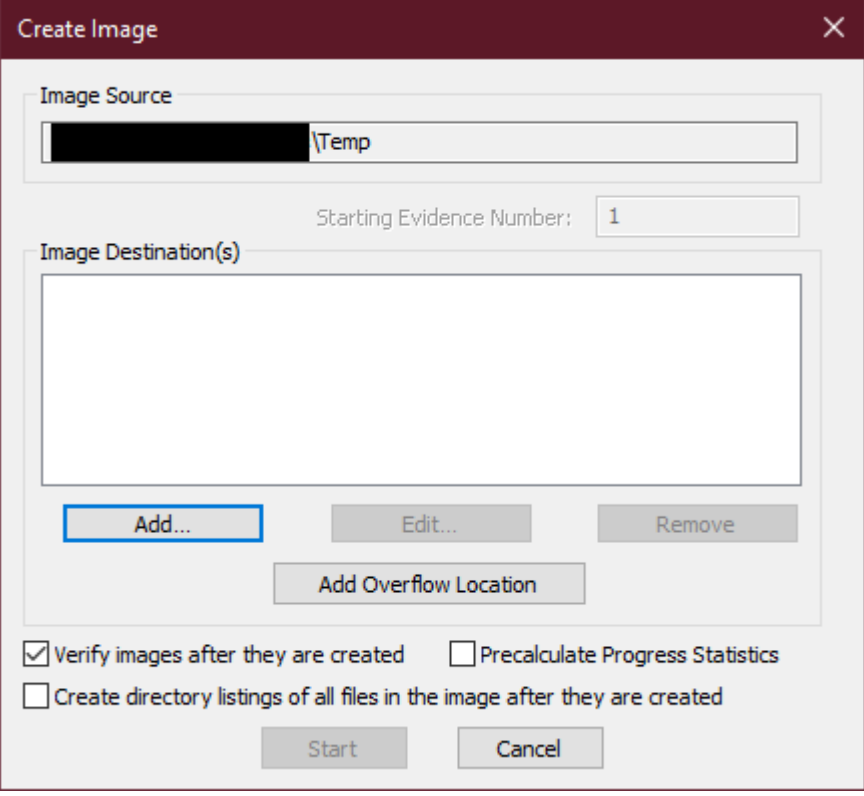
- The generated warning window can be ignored. Simply click on [Next](#).



- The window will now ask for a source location. Enter the location of your choice and click on [Finish](#).



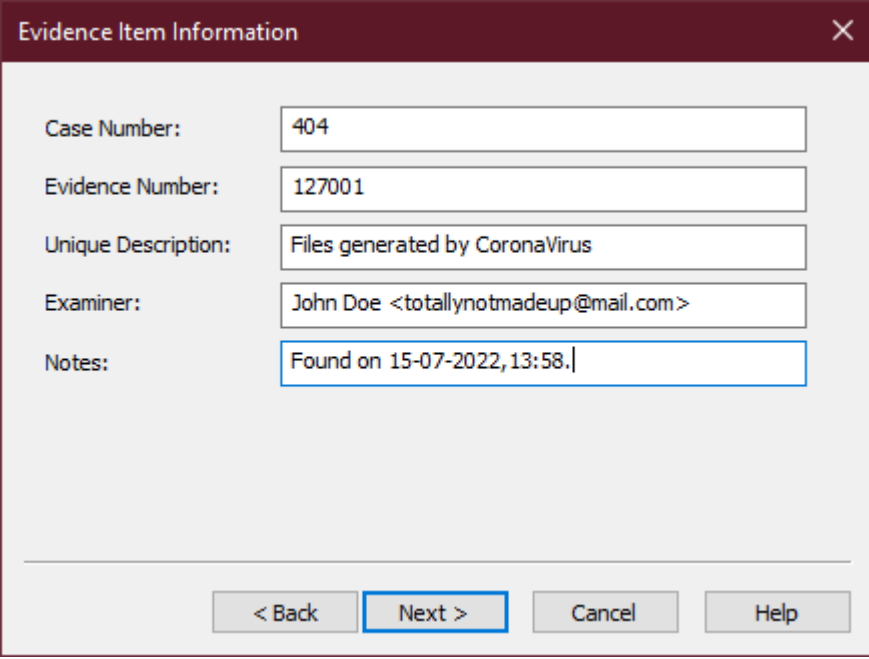
- Now, a new dialog box will appear. Confirm your source selection and then click on the [Add...](#) to add a new destination.



The 'Create Image' dialog box has a dark red title bar with a close button. It contains the following elements:

- Image Source:** A text field containing a redacted path followed by '\Temp'.
- Starting Evidence Number:** A text field containing the number '1'.
- Image Destination(s):** A large empty rectangular box.
- Buttons:** 'Add...' (highlighted with a blue border), 'Edit...', 'Remove', and 'Add Overflow Location'.
- Checkboxes:** ☒ 'Verify images after they are created', ☐ 'Precalculate Progress Statistics', and ☐ 'Create directory listings of all files in the image after they are created'.
- Bottom Buttons:** 'Start' and 'Cancel'.

- A new window will appear which will ask for information about this particular item. Fill it and then click on [Next](#).



The 'Evidence Item Information' dialog box has a dark red title bar with a close button. It contains the following elements:

- Case Number:** A text field containing '404'.
- Evidence Number:** A text field containing '127001'.
- Unique Description:** A text field containing 'Files generated by CoronaVirus'.
- Examiner:** A text field containing 'John Doe <totallynotmadeup@mail.com>'.
- Notes:** A text field containing 'Found on 15-07-2022, 13:58.' (highlighted with a blue border).
- Bottom Buttons:** '< Back', 'Next >' (highlighted with a blue border), 'Cancel', and 'Help'.

- Select the destination of your choice and provide the filename of the (soon to be) generated image file(s). Click on **Finish**.

Select Image Destination

Image Destination Folder

Image Filename (Excluding Extension)

Image Fragment Size (MB)
 For Raw, E01, and AFF formats: 0 = do not fragment

Compression (0=None, 1=Fastest, ..., 9=Smallest)

Use AD Encryption ☐

Filter by File Owner ☐

< Back **Finish** Cancel Help

- The newly created entry should now be visible in the **Image Destinations** list. Click on **Start**.

Create Image

Image Source

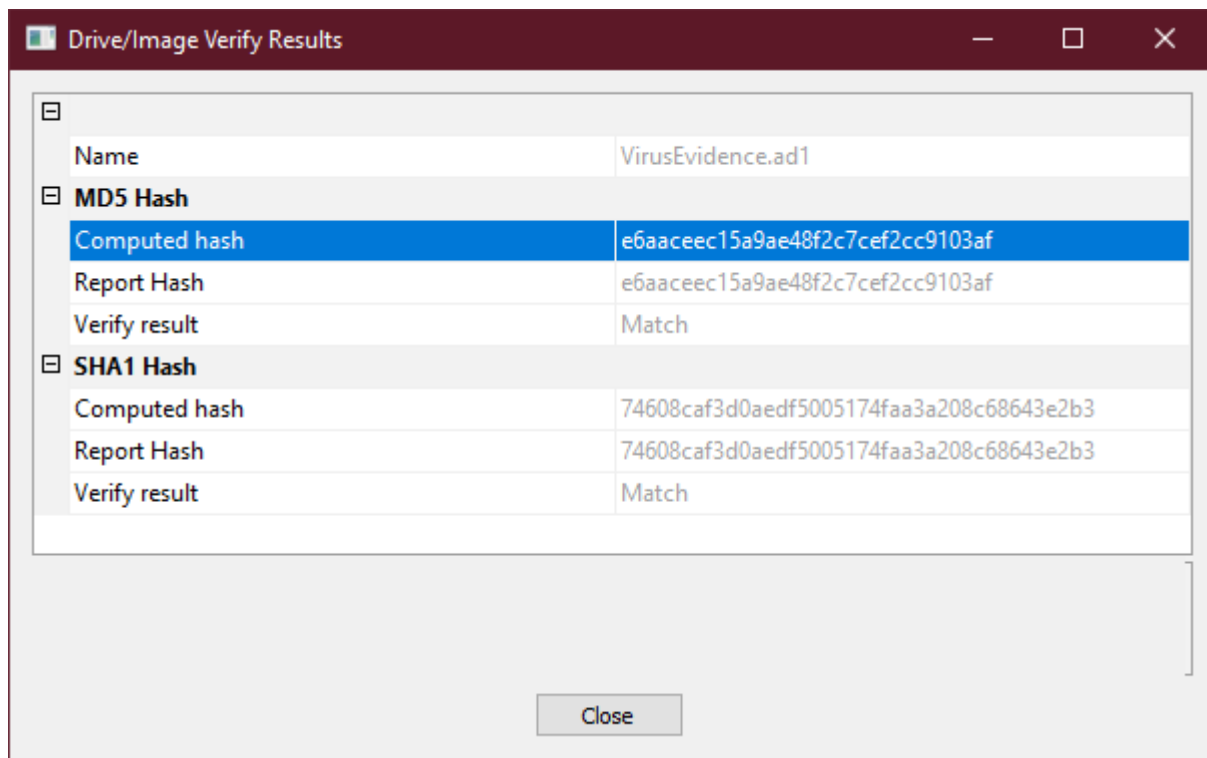
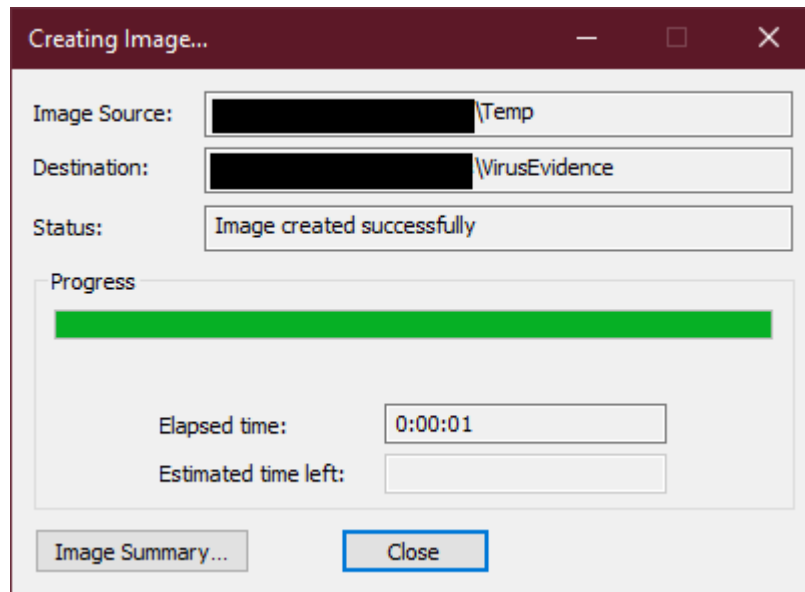
Starting Evidence Number:

Image Destination(s)



☒ Verify images after they are created ☐ Precalculate Progress Statistics
☐ Create directory listings of all files in the image after they are created

Start Cancel

- The process will take some time to complete (depending on the size and type of files/folders). After which you'll see a process completion screen and a verification screen.



- You'll also see some files generated in your destination folder.

Name	Date modified	Type	Size
 VirusEvidence.ad1	15-07-2022 14:01	Text Document	1 KB
 VirusEvidence.ad1	15-07-2022 14:01	AD1 File	17 KB

Practical 8

Aim: Retrieve deleted files from a computer.

Procedure:

- Download and install **Autopsy®** from this [link](#). Running the application should present you this window:



- Click on [New Case](#). It should present you this window asking for case name and the directory to store case-related data.

New Case Information

Steps

- Case Information**
- Optional Information

Case Information

Case Name:

Base Directory:

Case Type: ☒ Single-user ☐ Multi-user

Case data will be stored in the following directory:

< Back **Next >** Finish Cancel Help

- Enter the relevant details and click on **Next**. A new section will be available which will ask you to fill in optional information. You *may* choose to not enter any information in this section. Click **Finish** when you're done.

New Case Information

Steps

- Case Information
- Optional Information**

Optional Information

Case

Number:

Examiner

Name:

Phone:

Email:

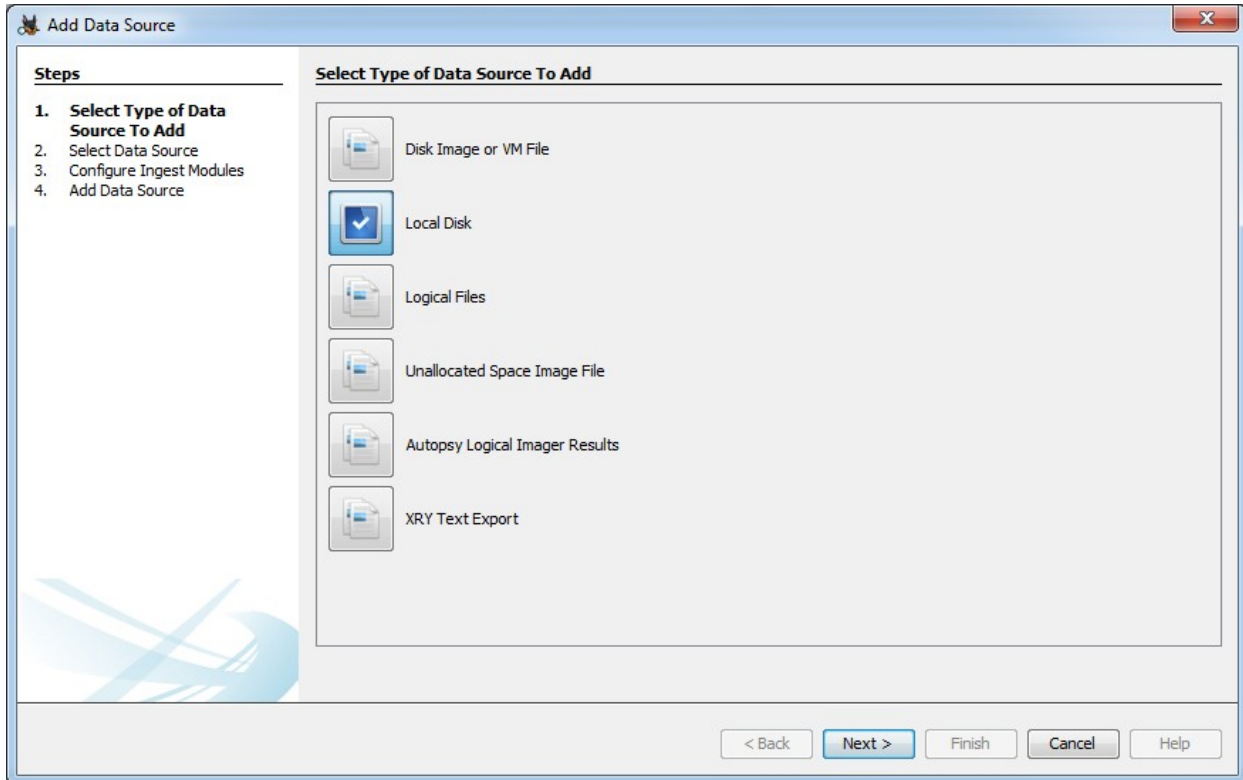
Notes:

Organization

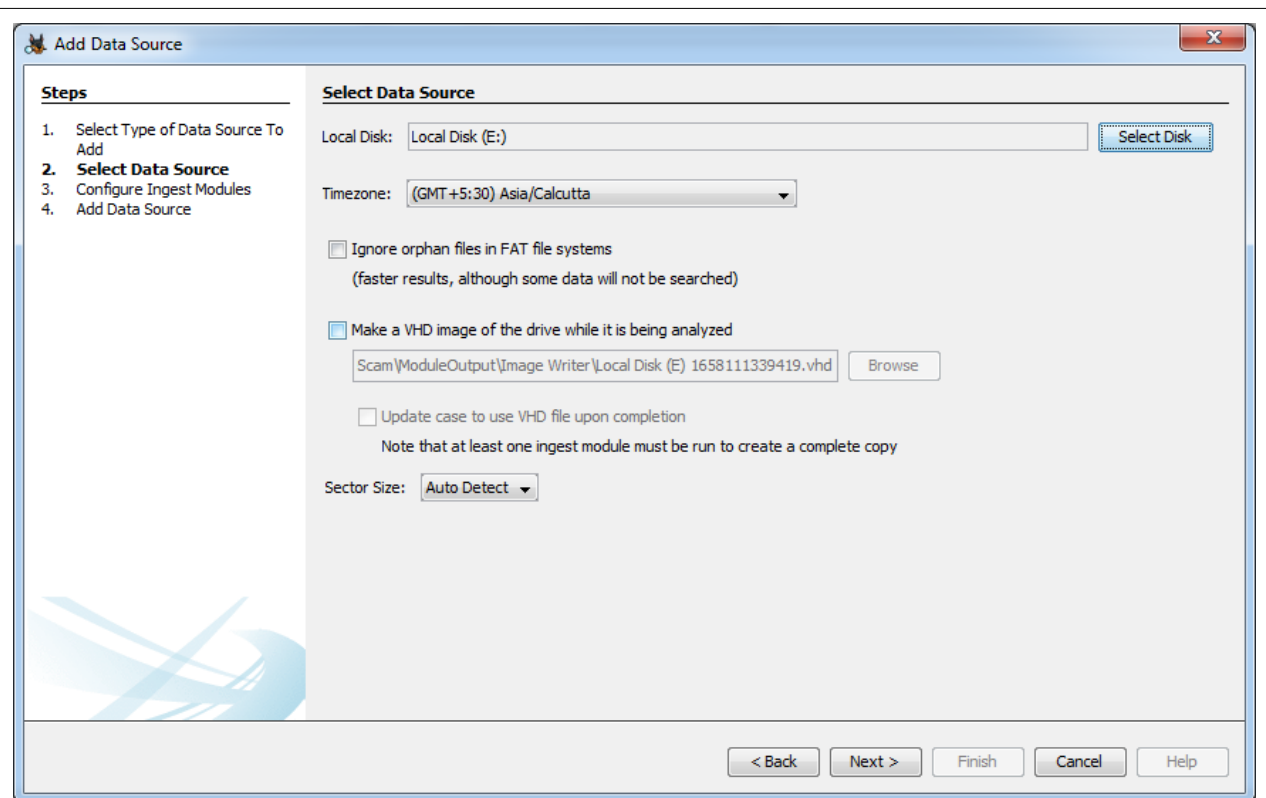
Organization analysis is being done for:

< Back Next > **Finish** Cancel Help

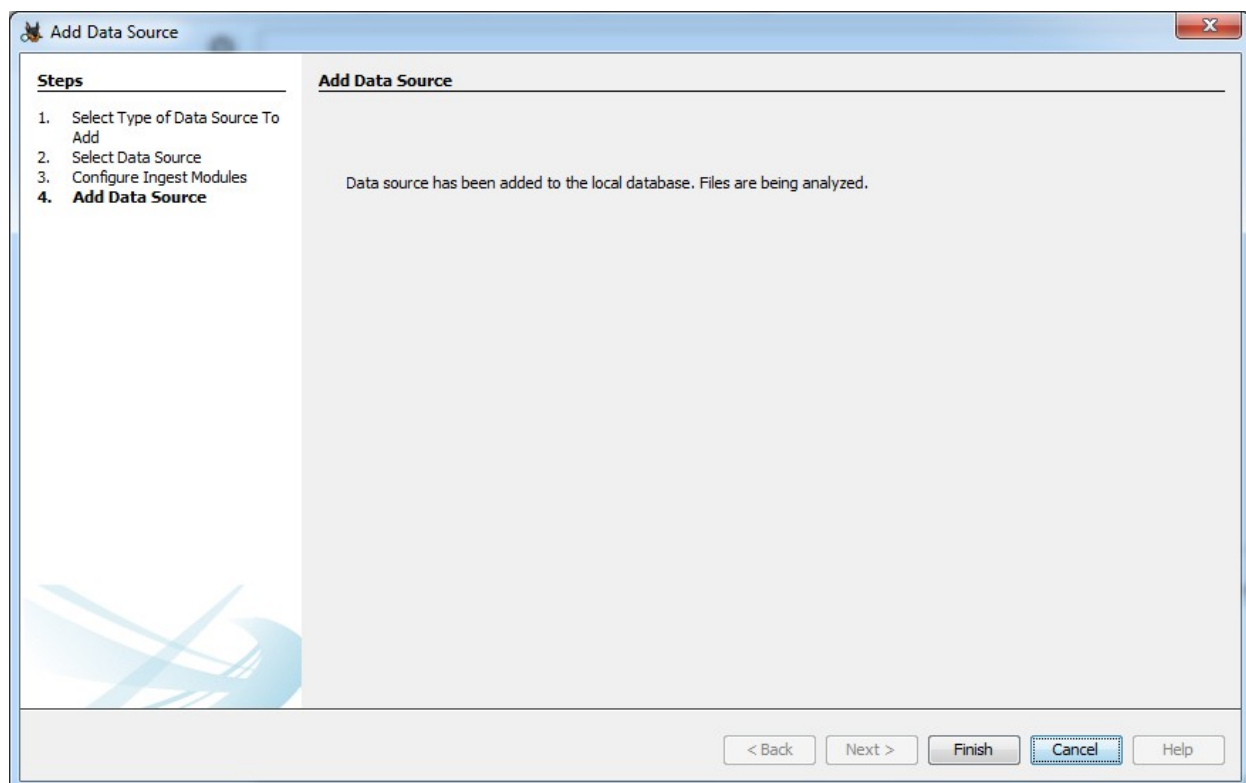
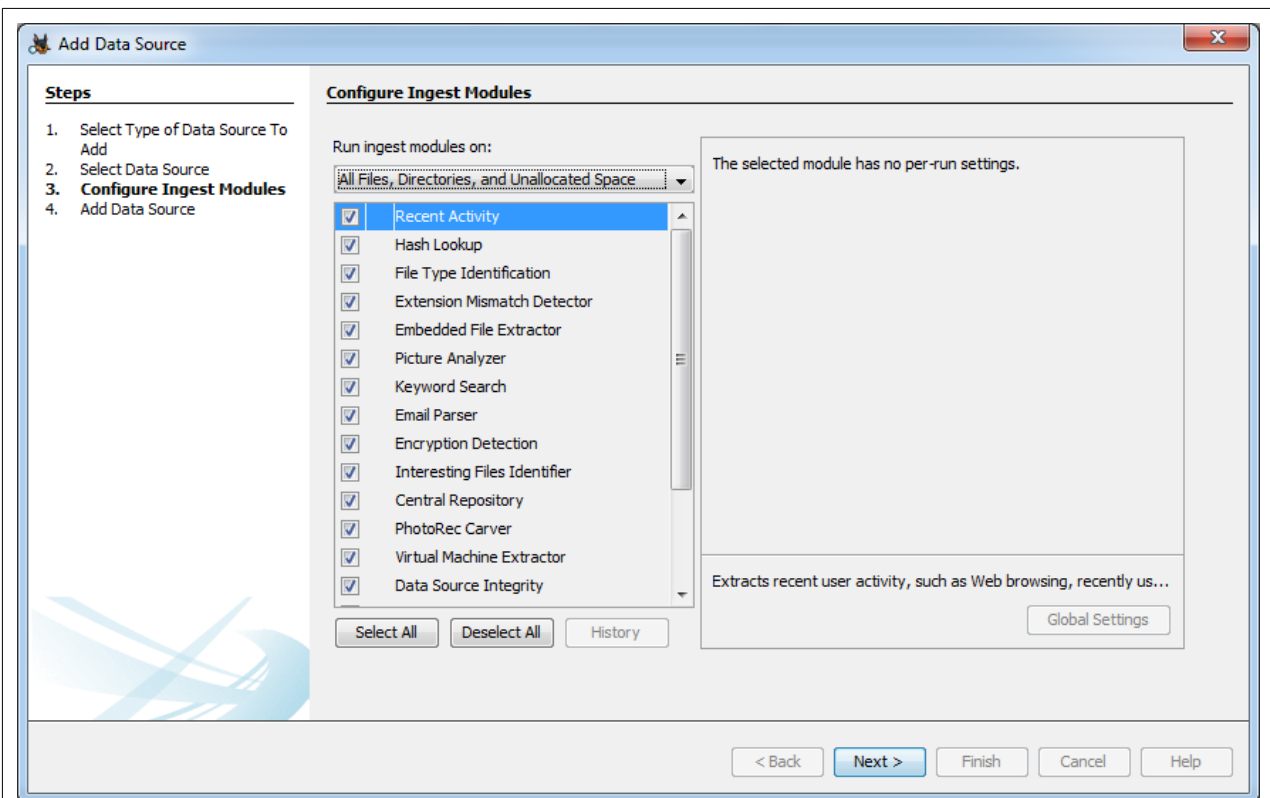
- A new window titled **Add Data Source** should now be visible. If it does not appear automatically, you can manually open it using the relevant toolbar item. Select **Local Disk** as the type of data source to be added and click on **Next**.



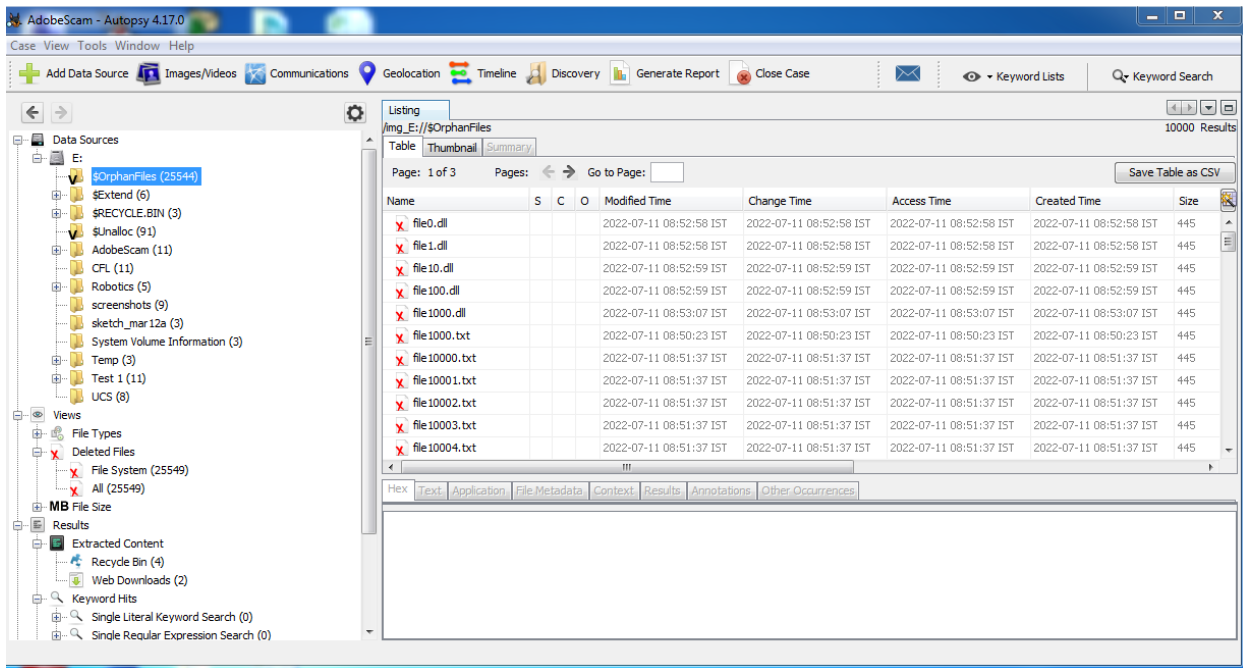
- A new section named **Select Data Source** should now be active. Select the disk of your choice and click on **Next**.



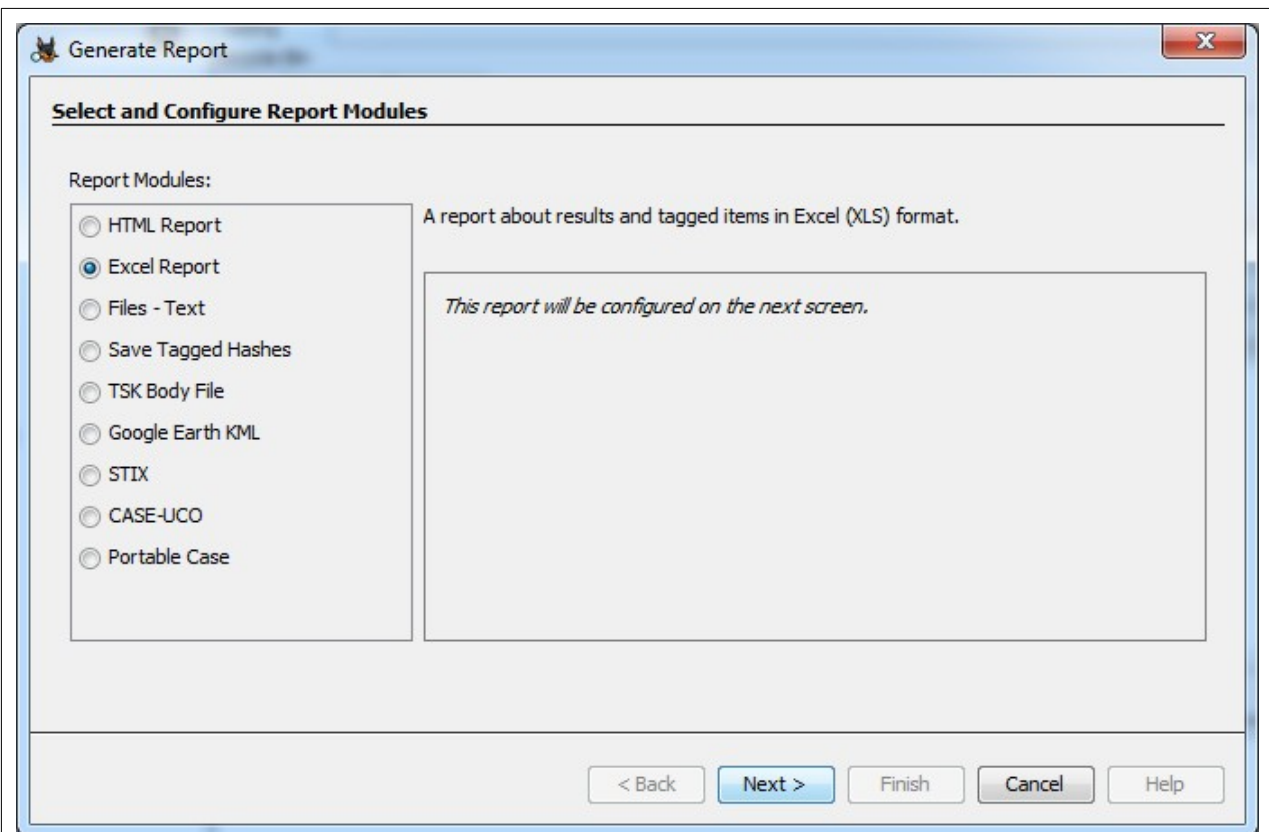
- You can use the default options in the **Configure Ingest Modules** section. After which, the data source will be added to the case database.



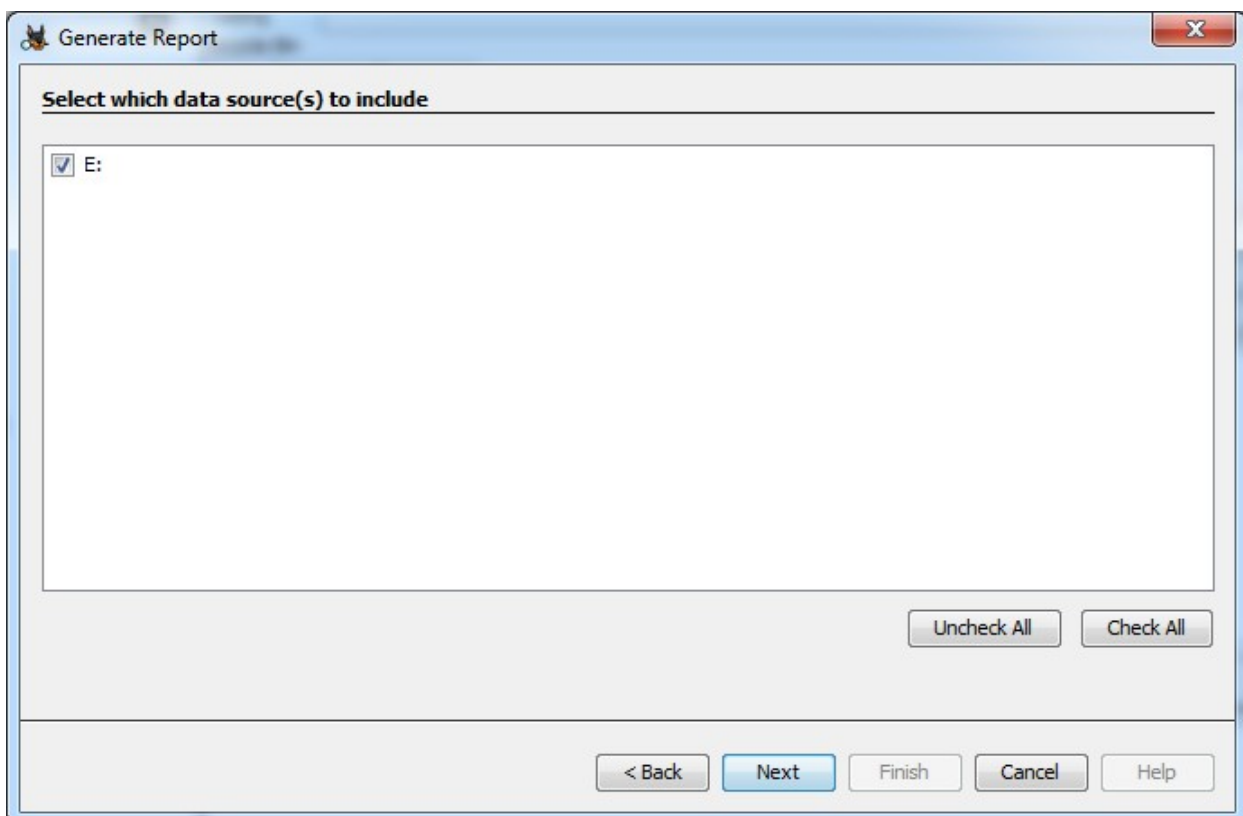
- **Autopsy®** will now try to process the data source. This process may take some time depending on the size of the disk and its contents. After completion, you will see all the information it has gathered ordered as a tree. Now, navigate to **Data Sources > {Disk of your choice} > \$OrphanFiles**. It will show all the deleted files. You can retrieve it by right clicking the file(s) and selecting **Export**. It will ask for a location to restore the file.



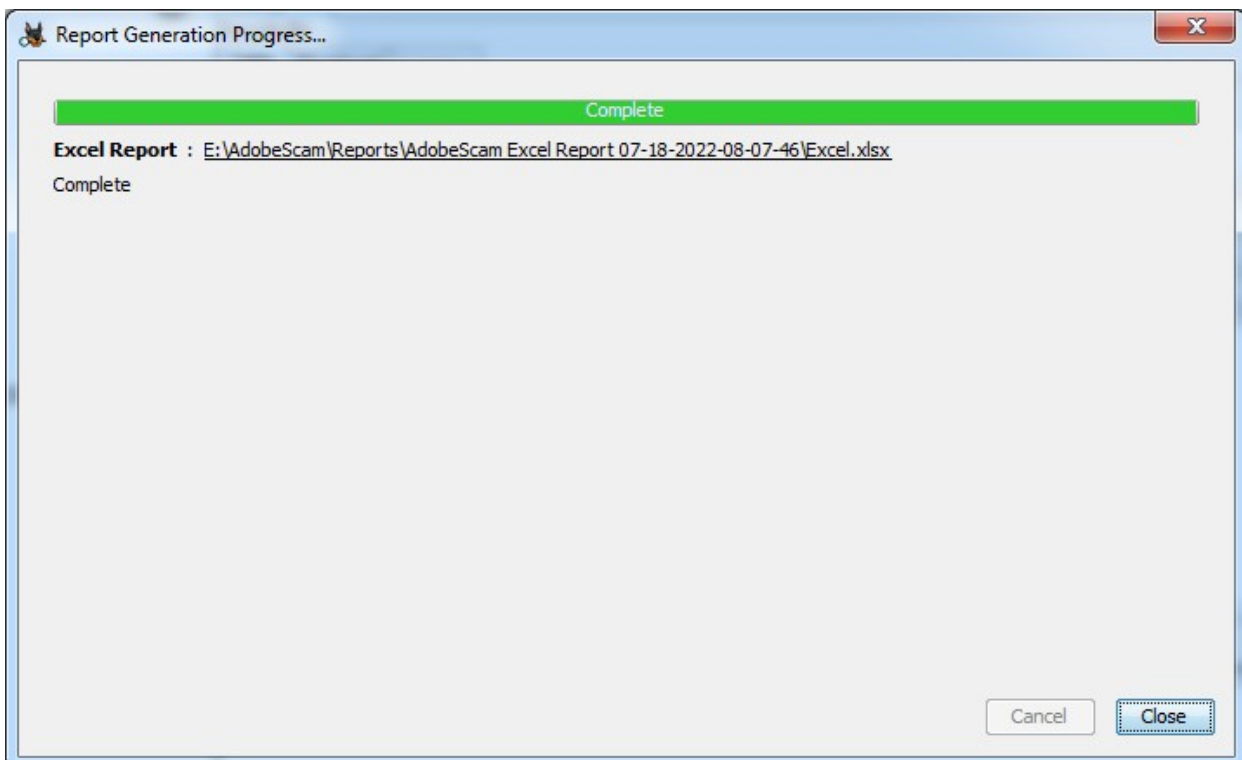
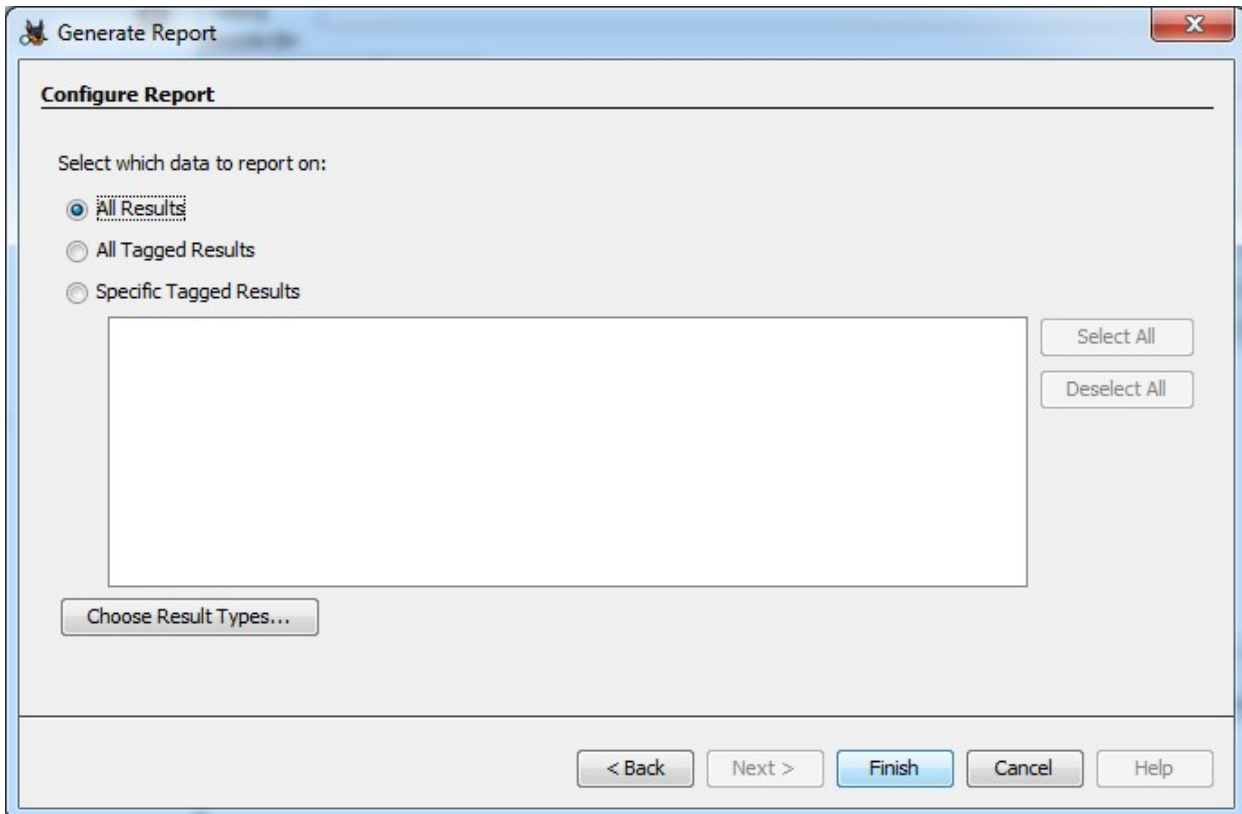
- To generate a report, click the **Generate Report** toolbar item. It should open a **Generate Report** wizard. Select the type of report you want and click on **Next**.



- Select the data sources to be included and click on [Next](#).



- Select the data which should be reported and click on [Finish](#). The report will be generated.



Practical 9

Aim: Use the registry to obtain information.

Theory:

The Windows Registry Editor(**regedit**) was launched in 1992 with Microsoft Windows 3.1. The registry is the backbone of the OS and is critical for system performance. It enables administrators and advanced users to keep the registry operational and make root and administrative level changes such as setting up access permissions or changing the hardware and software level configuration.

Features:

1. System Performance:

- If a key inside the registry becomes corrupt or faulty, it can cause system to crash or other performance issues.
- Using Registry Editor we can edit/update the key.

2. Configuration settings:

- The automatic type startup programs display or desktop setting can be configured using regedit.

3. Registry cleaning:

- Entries inside the registry can sometimes break. To fix broken entries, a registry cleaner is required.
- Unlike standard configuration files, entries inside the Registry cannot be opened or cleaned via standard text editor.

4. Registry errors:

- Certain events can disturb the hierarchy and cause errors.
- The **regedit** tool can be used to fix the hierarchical structure of the registry.

5. Finding Strings:

- **regedit** can be helpful when searching for specific strings in keys, values (names & values).

6. Remote editing of registry:

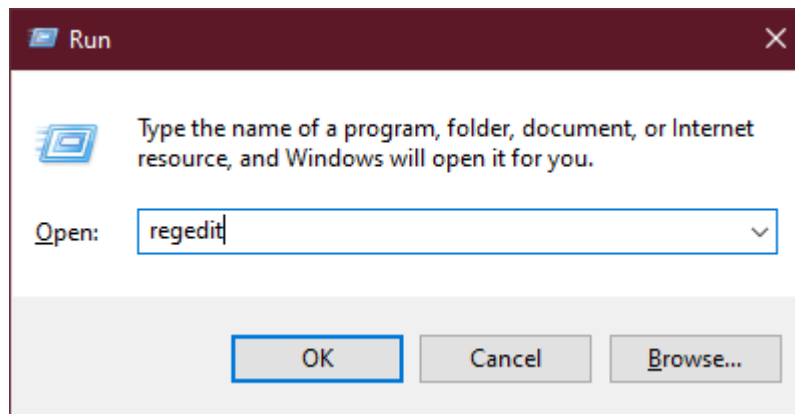
- **regedit** can be used for remote editing of another computer's registry on the same network.

7. Modification of key:

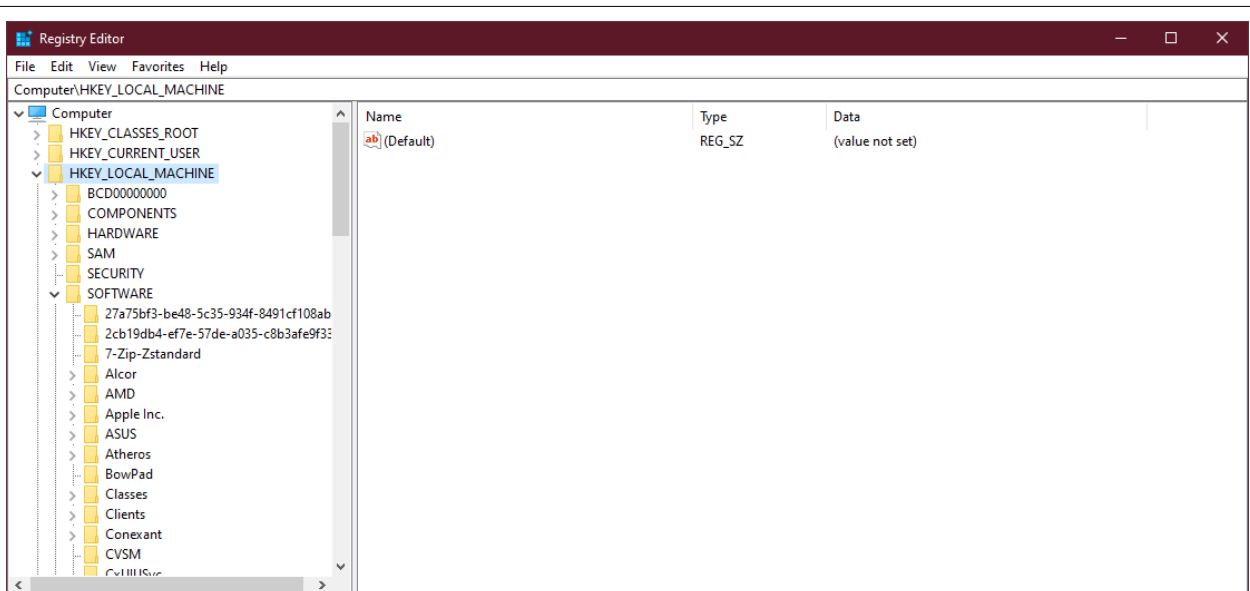
- Registry key can be modified, renamed or deleted by **regedit**.

Procedure:

- Press Windows key + R to access the **Run...** command.

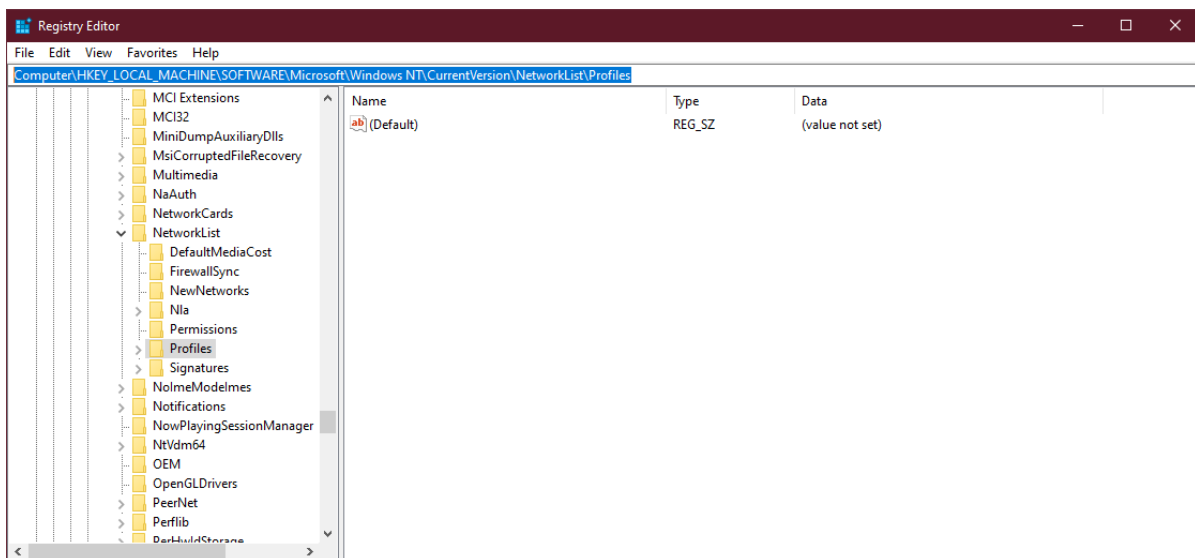


- Type **regedit** and press **[Enter]**.

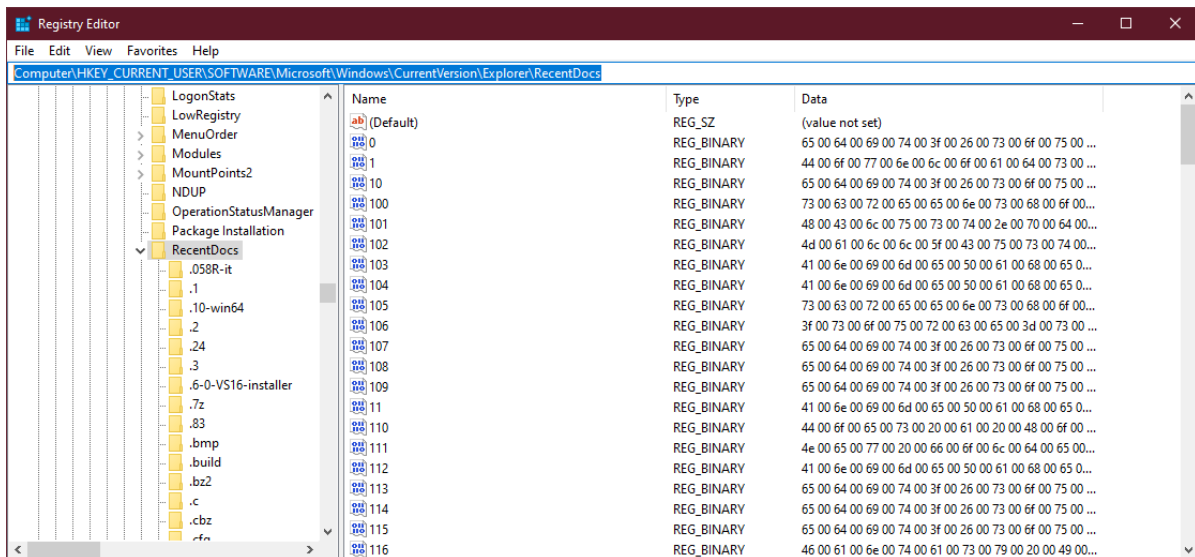


Locations:

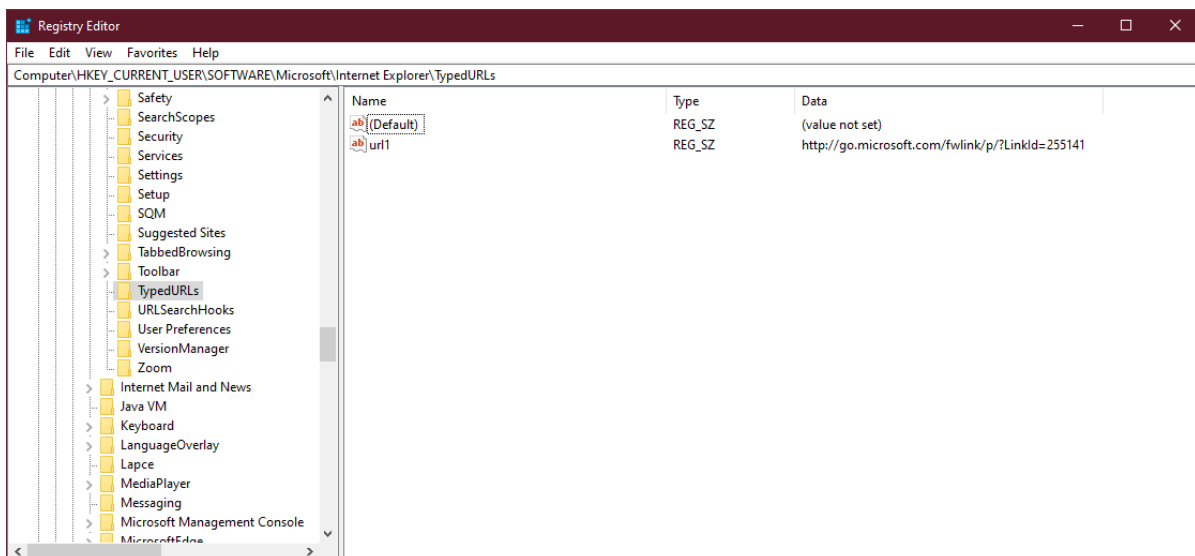
- Wireless Evidences: `Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList\Profiles`



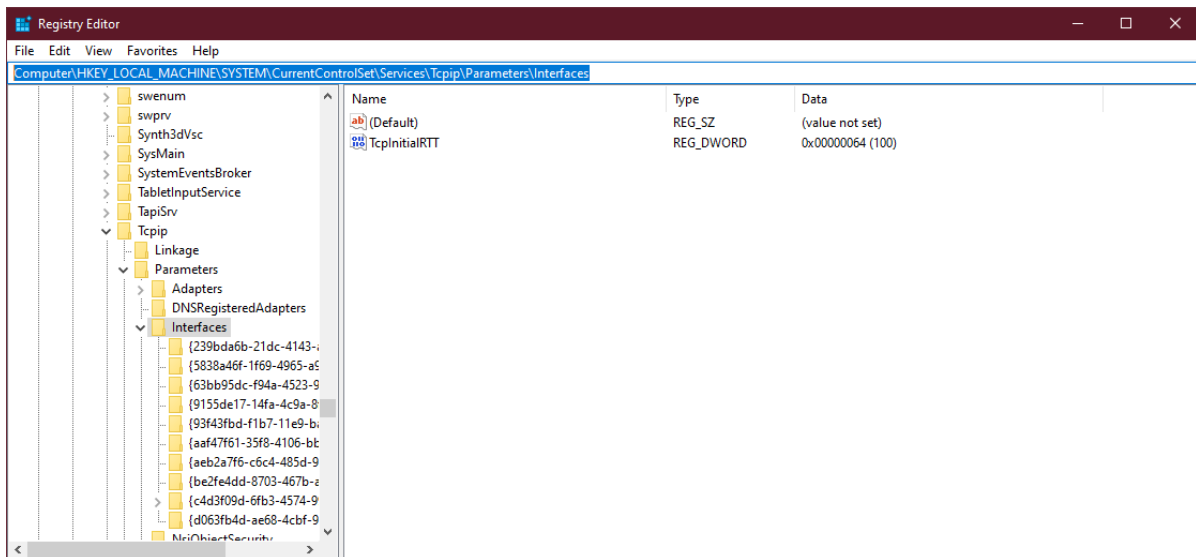
- Recent Documents key: Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs



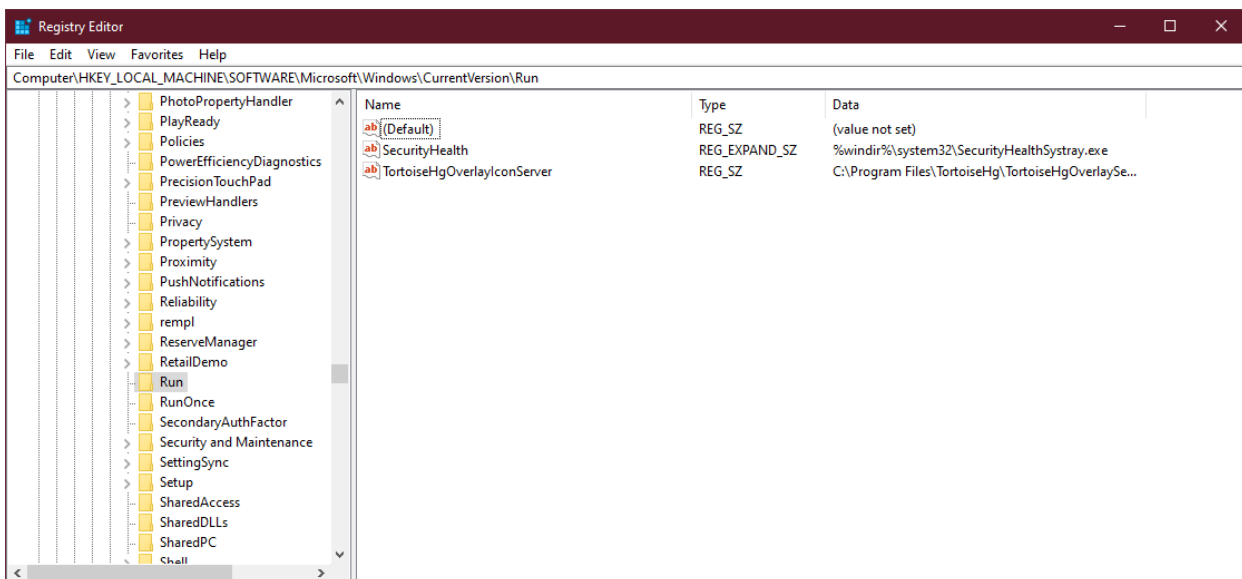
- Typed URLs key: Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Internet Explorer\TypedURLs



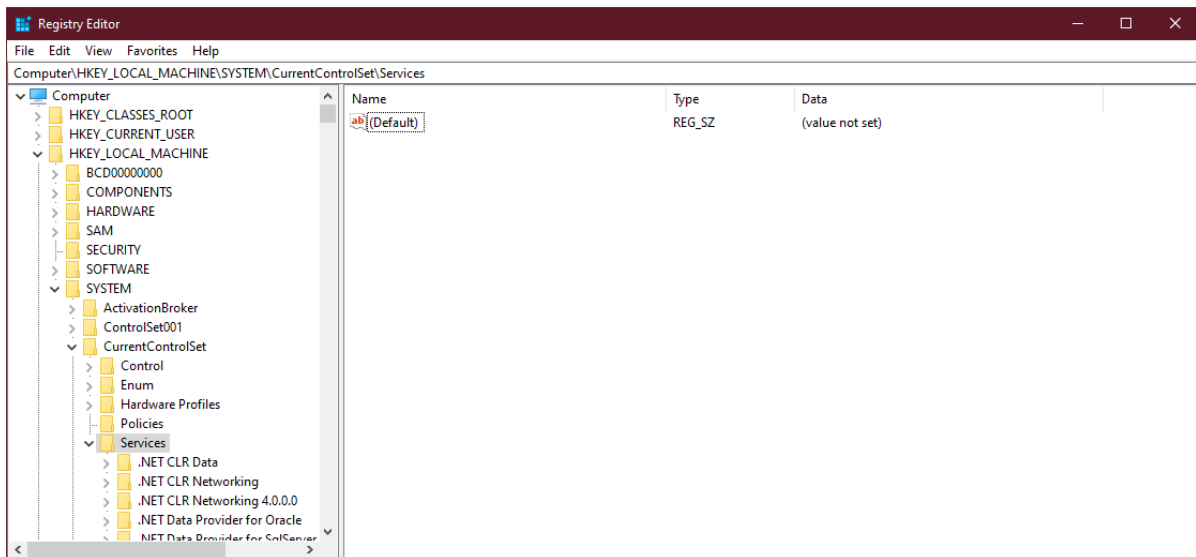
- IP address: Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces



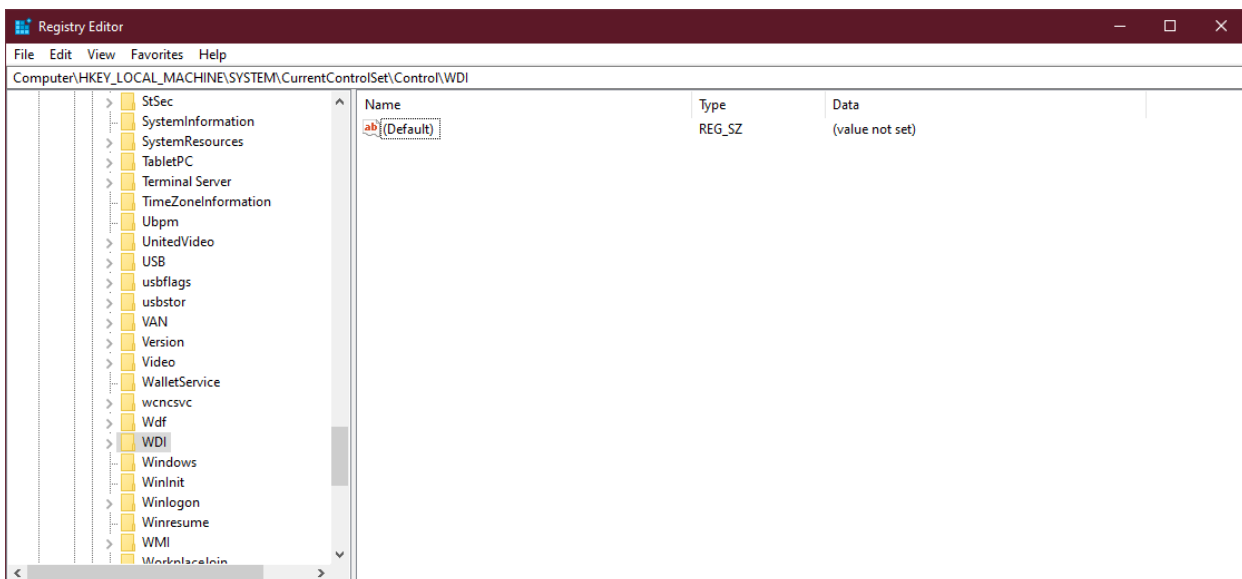
- Startup applications: Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run



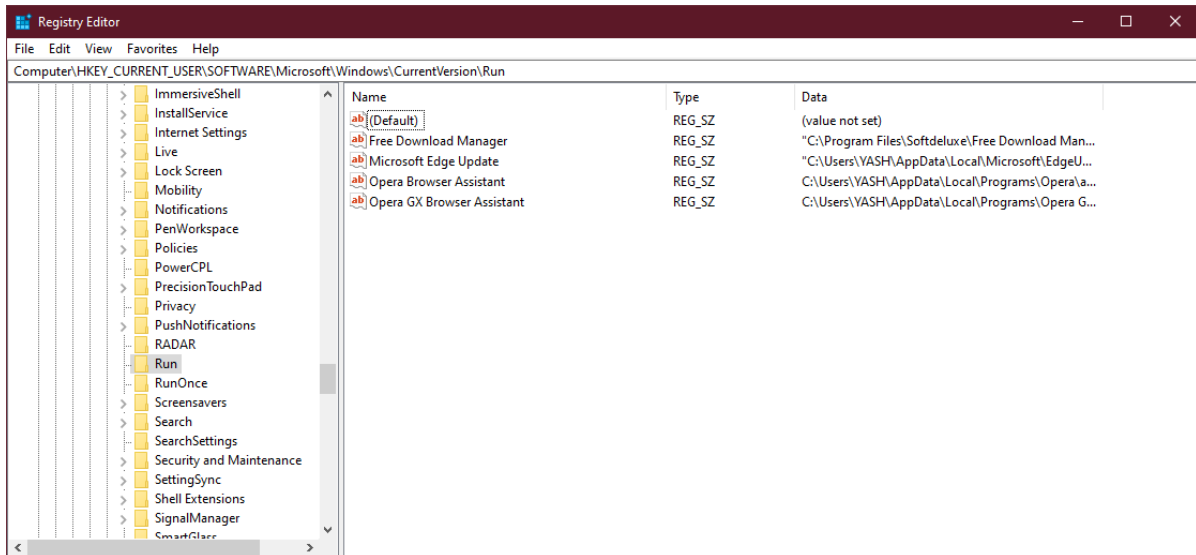
- Startup services: Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services



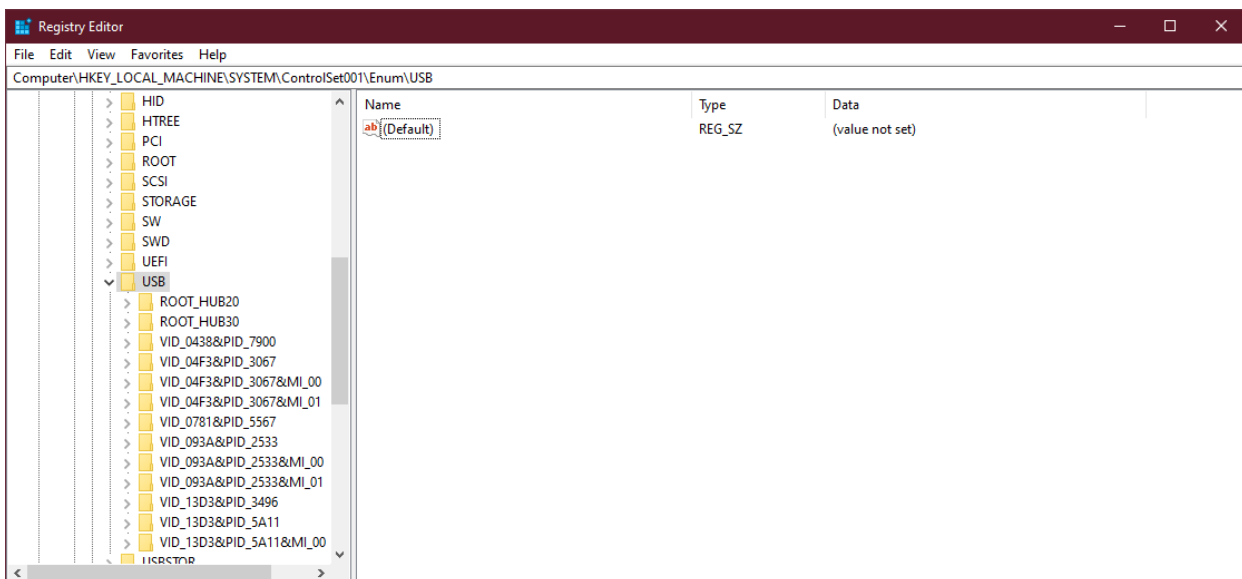
- Start legacy applications: Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\WDI



- Startup application(s) when a particular user logs in: `Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`



- USB drives: `Computer\HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Enum\USB`



- Mounted devices: Computer\HKEY_LOCAL_MACHINE\SYSTEM\MountedDevices

