CSCE 3304

Operating Systems

Disk Analyzer Report

Fall 2023

By:

Mohamed Abdelmohsen 900201290

Lina Serry 900203355

Frontend Implementation (Lina) :

Files: app.jsx, home.jsx, scan.jsx, styles.css

app.jsx:: sets up navigation and calls home

home.jsx:: displays home page and has button that links to scan

scan.jsx:: main function where implementation is done

styles.css:: file that has styling of all elements

Functions in scan.jsx::

saveDataToFile(filePath, sortedInfo):

- Takes file path and vector with the information and stores data to file, by invoking 'save_data_to_file' from the backend.
- Has error handling if 'filePath' is empty.

directory(path, chartType):

- Takes path and stores directory information into 'info' by invoking 'store_directories' from the backend.
- Uses path to save largest files into a vector 'largestinfo' by invoking 'scan_largest' from the backend.
- Handles chart buttons if pressed when the chart is already displayed.
- Sets 'currentPath' to given 'path', to be used later.
- Sets 'currentChart' to given 'chartType' to be used later.
- Error handling if path is empty.

deleteFile(path):

- Invokes 'delete_file' from the backend and gives it 'path'.
- Sets 'deleteSuccess' to true, to be used elsewhere, goes back to the parent directory using goToParentDirectory() then sets it back to false.

goToParentDirectory():

- Sets ‘parentPath’ to a substring of ‘currentPath’.
- Calls directory(parentPath) to process information of parent directory.

formatSize(sizeInGigabytes):

- Displays size unit according to size, whether KB, MB, or GB.

renderDirectoryData():

- Displays all files\folders in directory by name and size in a table.
- Sorts ‘info’ (output of directory()) by size and stores in ‘sortedInfo’.
- For each item in ‘sortedInfo’ it displays its name and size from formatSize(), and allows for each to be clickable, to display the respective directory.
- Displays a delete button if there are no children, else save directory button and placeholder for input are shown.

renderLargestInfoTable():

- Displays 5 largest files in directory by name and size in a table, only when ‘largest files’ button is clicked.
- Table disappears when button is clicked again.

aggregateSizesByType(info):

- Aggregates size of files of the same type, and returns an array with the total size of each type.

generateRandomColor():

- Randomizes colors for the pie chart, within a certain range of hues.

Scan():

- Main function

- Returns the display tables, buttons, and charts

Backend Implementation (Mohamed) :

Functions:

Save_data_to_file():

- This function provides a feature that does save every parsed directories into a txt file with its size in Bytes, Kilobytes, Megabytes, and Gigabytes.
- **Parameters():**
  - path: Path to the file where data is to be saved.
  - data: Vector of struct DirectoryInfo representing parsed directories.

Bytes_to_megabytes():

- Conversion function from bytes to Megabytes
- **Parameters:**
  - bytes - The size in bytes.

Bytes_to_kilobytes():

- Conversion function from bytes to Kilobytes
- **Parameters:**
  - bytes - The size in bytes.

Bytes_to_gigabytes():

- Conversion function from bytes to Gigabytes
- **Parameters:**
  - bytes - The size in bytes.

Delete_file():

- This function deletes the chosen file after analyzing the disk.
- **Parameters:**

    - Path to the directory I want to delete.

Calculate_directory_size():

- This function calculates every single directory that is being scanned.
- It calls the conversion functions to provide the appropriate size needed.
- It uses metadata to get the length of the directories.
- Returns a DirectoryInfo struct representing the calculated directory's information.
- **Parameters:**
    - Path to the directory.
    - indentation: String for indentation in the printed output.

Store_directories():

- This is the main function of the program
- It traverse through the directories using the Walkdir crate
- Inside this function, it has a file type breakdown, which checks on the last part of the directory string and strips it as its type.
- Vector of Directories is returned after pushing all the directories on the Disk to use in the Front-end.
- In each vector, Names, sizes, types, indentations(For Children nodes) are being pushed.
- **Parameters:**
    - Path

Scan_Largest():

- Using Rayon crate for parallelism, this function scans the directories to find what are the largest 5 files on the disk.
- A vector of largestFiles is returned to use in the Front-end.
- **Parameters:**
    - Path

Structs Used:

DirectoryInfo struct:

- **This struct is used in every functionalities that require the directories attributes:**
  - Path string
  - Size in bytes
  - Size in kilobytes
  - Size in megabytes
  - Size in gigabytes
  - Indentation
  - Name of the file/folder
  - File type

LargestFileInfo struct:

- **This struct is used in the Scan_largest Function which require the attributes:**
  - Path string
  - Size in bytes
  - Size in kilobytes
  - Size in megabytes
  - Size in gigabytes
  - Name of the file
  - File type

SaveFile struct:

- This Struct is being used in the save_to_file function where it uses those attributes: a boolean, and a String

Crates Used:

- **Std** crate => no version number (implemented directly in rust libraries), it has two functionalities we used:
  - OpenOptions : open a file to do an action
  - Write : write the directories in that file
- **Walkdir** => version: 2.3
  - It is the main crate we used for efficiently walking through a directory tree.
  - It allowed us to iterate over the entries in a directory and its subdirectories.
  - Useful for scanning the largest files alongside Rayon..
- **Serde** = { version = "1.0", features = ["derive"] }
  - JSON data structure
- **Serde_json** => version: 1.0
  - It allowed us to work with JSON data in rust
- **DiskScan** = > version : 0.3.0
  - Used it for scanning and analyzing disk information.
- **DiskExt, DiskKind, System, SystemExt** (Part of SysInfo Crate , version: 0.29.10)
  - Used them to find info about the disk as well such as name full storage, used storage, and available storage.
- **Tauri** => version: 1.5.2
  - Tauri is a the framework we used for building our disk analyzer
  - It allowed us to create the Front end logic using HTML, CSS, and JavaScript.
- **Rayon**=> version: 1.8.0
  - Was Useful for parallelizing disk analysis tasks for improved performance in the largest files scanning

Features Implemented:

- Scanning the whole disk
- Traversal through directories, by pressing on the name to move inward and a button to move out
- Display of the current path as a header
- Listing all directories inside disk with their sizes and respective units
- Scanning a specific directory
- Sorting the directories by size as a default
- Scanning the largest files and displaying the 5 largest ones of each directory (button to show and hide)
- File type breakdown
- Visualization of file type breakdown in a pie chart (button to show and hide)
- Pie Chart visualization of the scanned directories (default)
- Bar Chart visualization of the scanned directories (button to show and hide)
- Tree-Map visualization of the scanned directories (button to show and hide)
- Delete files from inside analyzer
- Save directory information into a txt file

Use Cases:

The user opens the app to the home screen, with the button ‘start scanning’ that when pressed on scans the whole disk. The directories of the disk are displayed, sorted by size, along with the corresponding pie chart. The user then can choose whichever directory to scan further by pressing its name, and go back to a previous directory by pressing the ‘back’ button on the top left. By doing so, the list of directories inside replaces the current list, and a pie chart is rendered by default. Other chart options are available by pressing the corresponding ‘bar chart’ and ‘treemap’ buttons. Hovering over sectors of the charts shows the name of the directory and how much space it is taking up. The buttons work to display and hide the charts. To display the largest files in the directory the button ‘largest files’ can be pressed, which displays a list of the 5 largest files along with their sizes. The list can also be hidden by re-pressing the button. The ‘file type breakdown’ button displays a pie chart with each sector showing a type of the files in the directory and the percentage of space each occupies, hovering over the sectors shows the space in bytes. Pressing the button again hides the chart. On the top right, there exists a placeholder to enter a specific directory path to scan, along with the button to start scanning it. If the user doesn’t enter a path an error message is displayed. Under the list of directories a placeholder is found which takes the path of a file to save the current information displayed to. A message is also displayed if no path is entered. If the user wishes to delete a file they would press on it and a delete button appears. By pressing delete, the file is deleted forever, a confirmation message is displayed and the program goes back to the parent directory. On the top left, ‘home’ is displayed, and when pressed on, it takes the user to the home page.

Through our survey, we found that each existing analyzer had unique features, so we tried to combine as many of them as we could to achieve the most functionality. By including different viewing options, file type breakdown, largest files, deleting files, and saving results in a pleasing GUI, we believe that we have created an analyzer that is very capable and can heavily compete with the existing ones.