CSCE3401 Operating Systems

The American University in Cairo

Project I

Disk Usage Analyser

Final Project Report

Fall 2022

Authors

Jana Shalaby 900201230

Omar Fayed 900191831

Zeyad Usama 900193027

Table Of Contents

Abstract

Disk analyzers operate one of the most important functions in the operating system. Due to the importance of disk analyzers this project was designed and modelled for an efficient disk analyzer .Our disk analyzer is based on previous disk analyzers that we studied and analysed before. This disk analyzer combines many functionalities of available disk analyzers in the market. Moreover it combines all efficient features that make it more sufficient than any other disk analyzer. In this report we will be discussing the implementation , design and functionalities of our disk analyzer , In addition to adding screenshots and pictures of its informative GUI.

Project Overview

Our project is to design and develop a disk analyzer for Linux in Rust. Its purpose is to scan your file system structure and graphically represent the directory entries with emphasis on size occupied by each entry.

Project Plan

After reviewing and discussing the four different disk analyzers in our last survey , we decided on a layout and features that we want to implement on our own disk analyzers. The disk analyzers that we reviewed allowed us to visualise the options and GUI we feel comfortable in using and developing in our disk analyzer. Regarding the implementation of our own file disk analyzer , we were aiming to implement a unique GUI. For instance in the survey as discussed before the disk analyzer Qdirstart , it offers a very helpful function which allows the user to view the files space in progress bars and also allows for functionalities such as cleaning and deleting files to be done directly from within Qdirstart software. On the other hand, the disk analyzer *DUtree* provides the user with a tree view of files and directories. One of our goals was to implement GUI based on pie chart classification and organisation rather than blocks. For example, a multilayer pie chart where users can search for a specific directory by selecting the pie-chart segment that corresponds to that directory. For choosing the directory or file to analyse.

Fortunately , we have created and implemented all the functions we stated in our project plan in addition to adding more helpful functionality and enhanced GUI. We have created a fully functional disk analyzer that contains many features of the viewed disk analyzer from our last survey. We have included the pie chart GUI that we viewed in *Filelight KDE and DiskSavvy* .Moreover we have also implemented the space progress bars of files inspired by the GUI of *QDirStat.* Also we have managed to implement the same deleting and cleaning files functionality as *QDirStat.* Another informative feature we implemented is to display the files and directories in a tree form view and that idea was inspired by *DUtree.*

Projects Phases

- *Phase 1*

In Phase one , we started to look more into rust , watching tutorials and learning all the skills to master the programming language. It took us a while to learn Rust as it's a quite new programming language with a small community as well as it is classified as a totally new paradigm from the programming language paradigms we generally use.

- *Phase 2*

In Phase two , we did a survey on most of the file disk analysers available online and in the field of disk analyzer. We also created our project plan and collected all the ideas inspired from the disk analysers we surveyed.

- *Phase 3*

In Phase three, we started implementing the backend function using Rust. we implemented all our disk analyser features in functions and texted them.Implementing the delete , create and search by extension functions

- *Phase 4*

In phase four , we created our disk analyser GUI . we starting integrating the backend function we created with the frontend code and libraries.Implementing the file tree, pie chart , bars and progress bars

- *Phase 5*

Phase Five was our last phase , we created our project report , tested our disk analyser with all possible scenarios and situations on how the user can use it and how it outstands from the rest of all other disk analyser

Disk Analyzer Functionalities

- *Analysing Directory*

As our disk analyzer starts operating , it prompts the user for the directory to analyse. The user has two options for input . The first option is to drag and drop the folder to be analysed.The second option is to write the folder full path to analyse.

- *Displays Directory in a Tree Form*

The disk analyser then takes the input directory form the user and displays all the files in the directory as well as all the sub directories in a tree form view.

- *Displays Directory in Blocks*

In addition to displaying the files in a tree sequence , all the files in the directory are displayed in the form of blocks , where there are two columns of blocks .One column contains the name of the file , another contains the size of that file. Both columns of blocks are placed next to each accurately equally spaced.

- *Displays Directory in Pie Chart*

Directory files are displayed in the form of a multi colour pie chart. The pie chart contains all the files in the directory ,each labelled,coloured differently and each section size is according to file size.

- *Displays Directory files sizes progress bar*

Directory files can be viewed in a form of progress bars according to their sizes.Each progress bar displays the percentage of file size from disk.

☐ *Creating New Files & New Directories*

Our disk analyser enables the user to create new files and new folders in the directory they choose.

☐ *Deleting Files & Directories*

Our disk analyser enables the user to delete and clean files and folders in the directory they choose.

☐ *Search Files by File Extension*

Using the disk analyser the user can search and filter through the files by file extensions.

Disk Analyzer GUI

Implementation of our disk analyser GUI enabled us to create a very user friendly , easy and informative user interface. We viewed many other disk analysers and searched more into their GUI libraries.Examples we looked at GTK , FlightLight and more. Lastly we have decided to use FLTK libraries as well as Fluid libraries. It was a very helpful tool to integrate with using Rust.

Using the FLTK libraries , it helped us create the window widgets and objects . FLTK enabled us to create widgets for each component we wanted to display. Moreover it allowed us to create buttons for each function , for instance analysing a directory , we create a button for the user to press in order to call the function and analyse the directory.Also view the directory files in a tree form was created by implementing and integrating a build in class and library in FLTK that declares and create a tree of directories , files and subdirectories. Another important feature FLTK provided us to use is the colour themes for our disk analyser , Using the colour themes libraries in FLTK enabled us to create a good looking and bright disk analyser GUI.

Furthermore , Using Fluid libraries enabled us to have control on the widgets and window sizes and dimensions of our GUI. Resizing widgets and windows helped us in creating a well sized and precise dimensioned GUI for our disk analyser. Also we were able to change the location of each widget on the window using Fluid Libraries .One more important feature we created using Fluid libraries is the option for the user to drag and drop folders to analyse.

FLTK and Fluid libraries offered us a visualisation of the GUI we are creating and that was an important factor of creating a clean GUI as we edited , changed and enhanced it as we were implementing the code .

Data Structure used

Vectors :

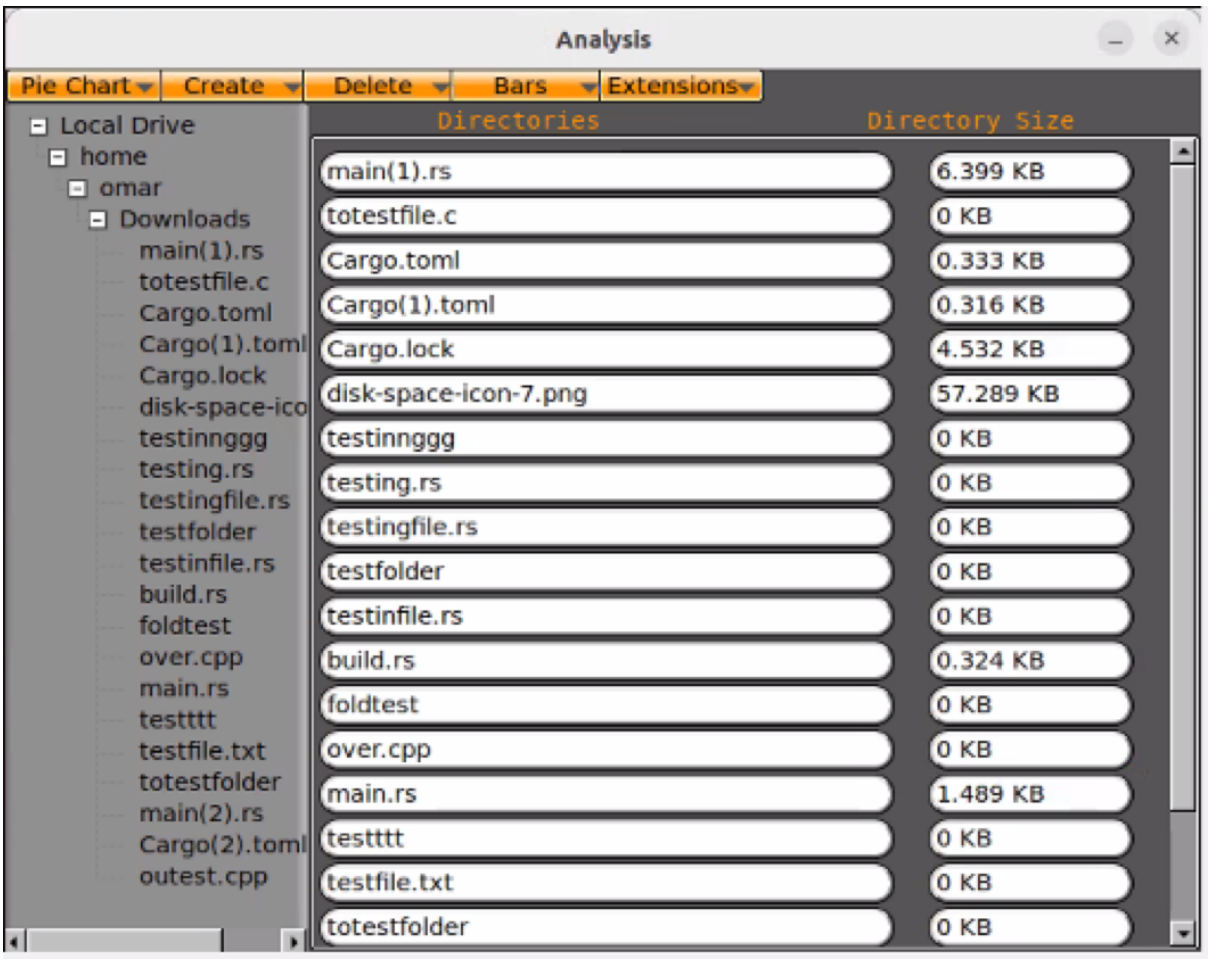
We depended in our code on vectors in multiple parts of the code. First we stored the files and folders names and their sizes in vectors with corresponding indexes to print them in correct order. The same feature of vectors we used in getting file extensions and to print them to the user.

Corresponding sizes were stored in a vector so that the percentage of space consumed by files and subdirectories were presented correctly in the progress bars as well as accurately presenting the proportions consumed by each file/directory in the pie chart. Storing individual file sizes was important in calculating the percentage of space consumed for a given extension. This was part of the searching by extensions feature.

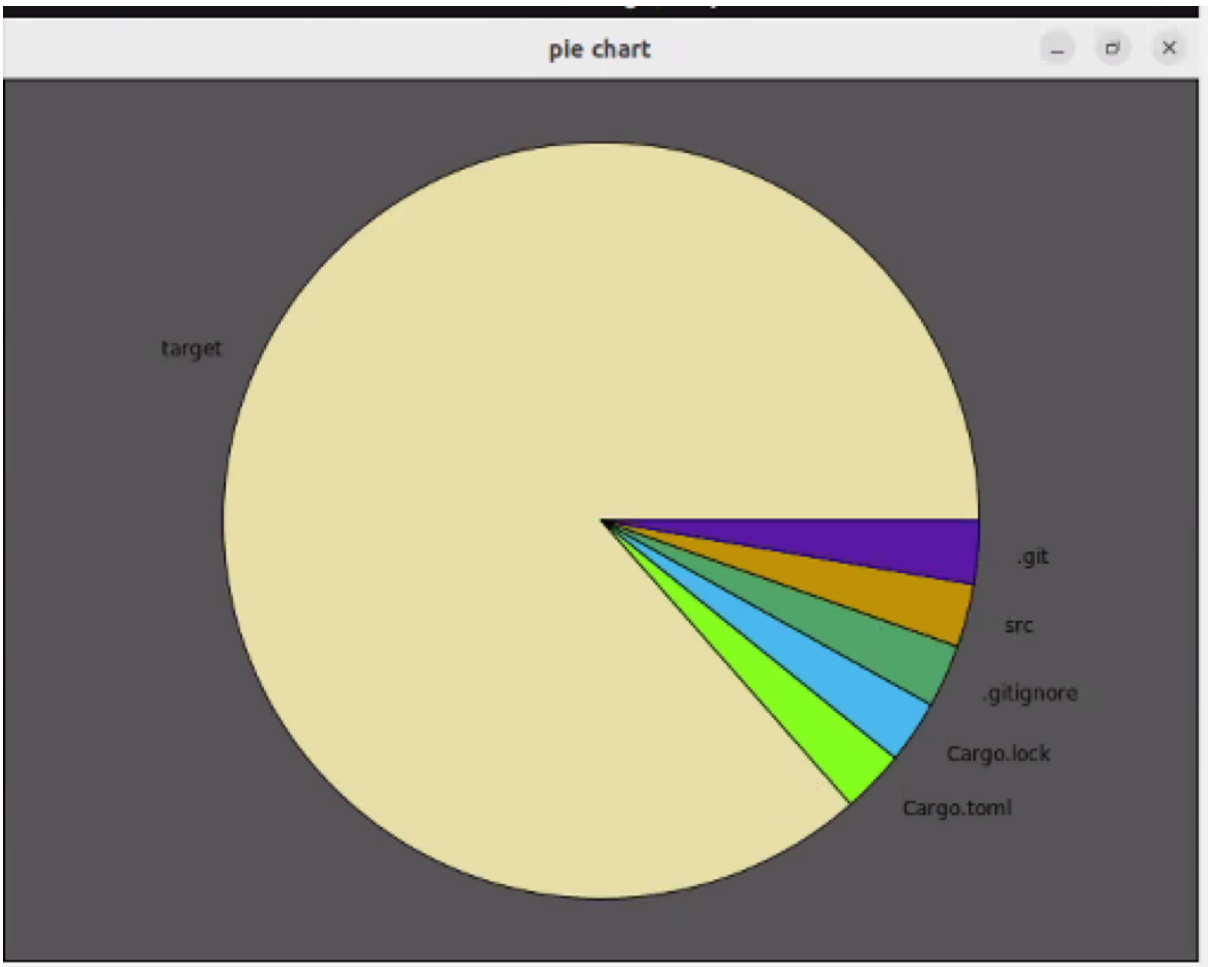
We used other generic data types for the rest of the code without using special or extra builtin functions.

Screenshots

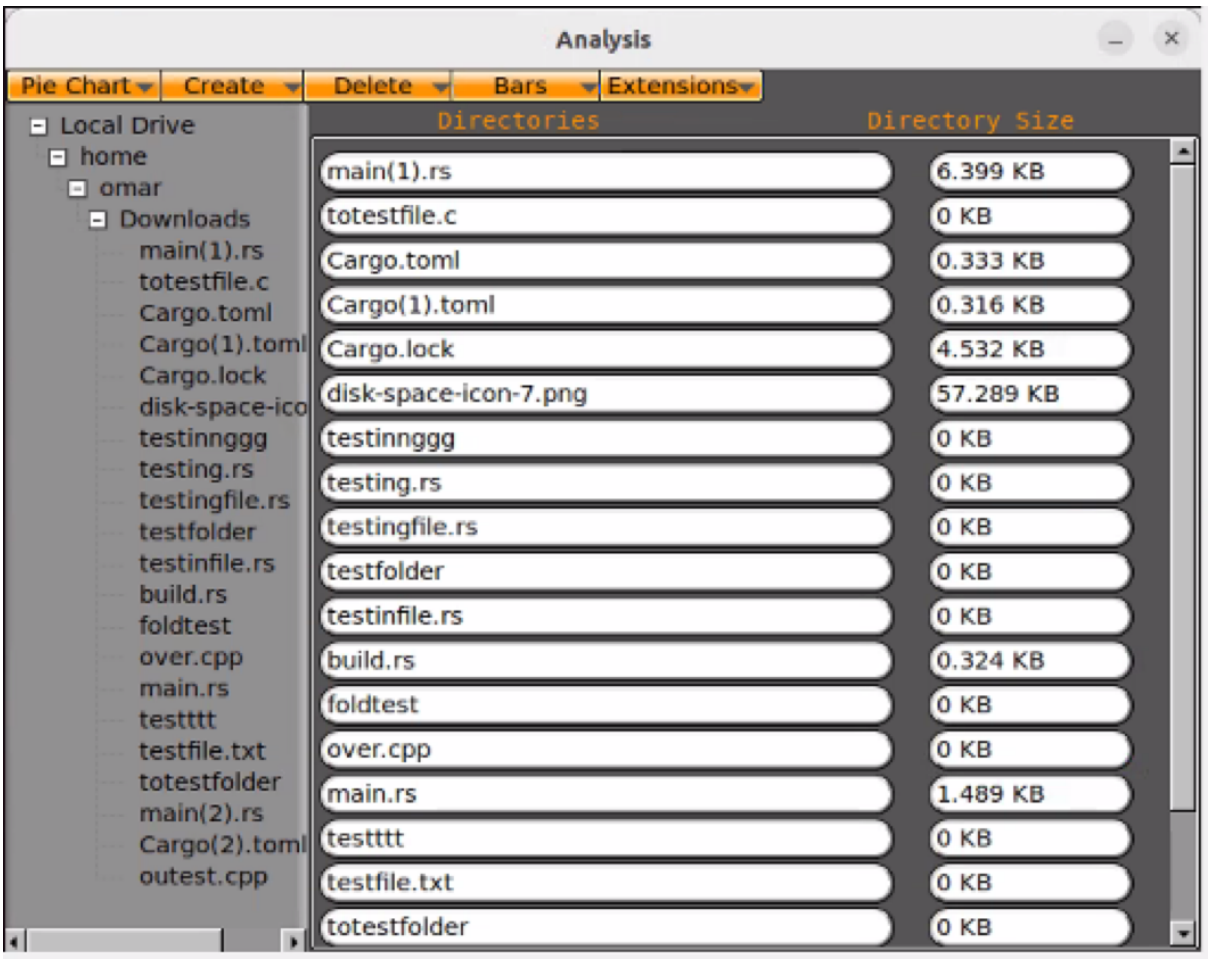
Enter Folder name to analyse



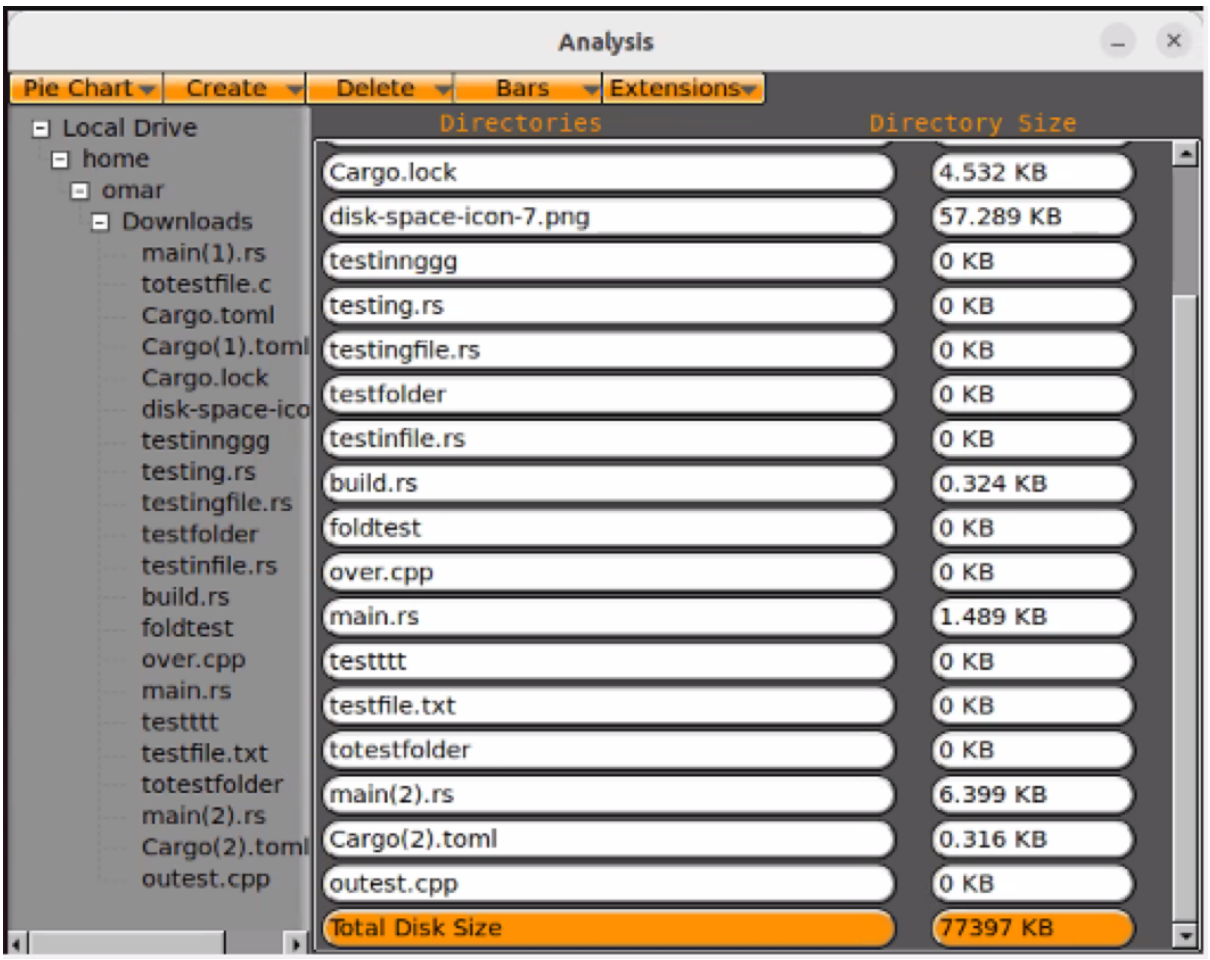
Analyzing folder



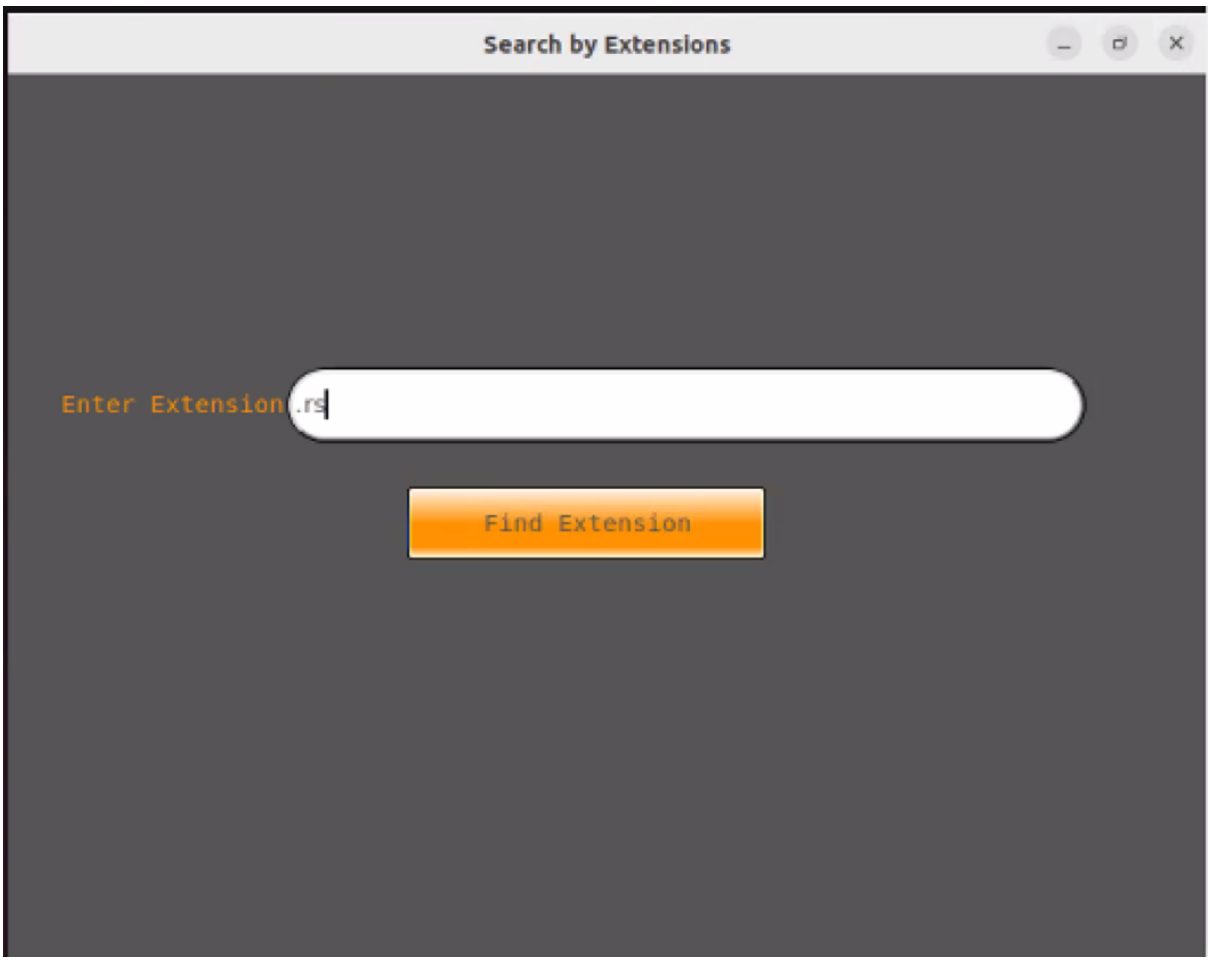
Pie chart View



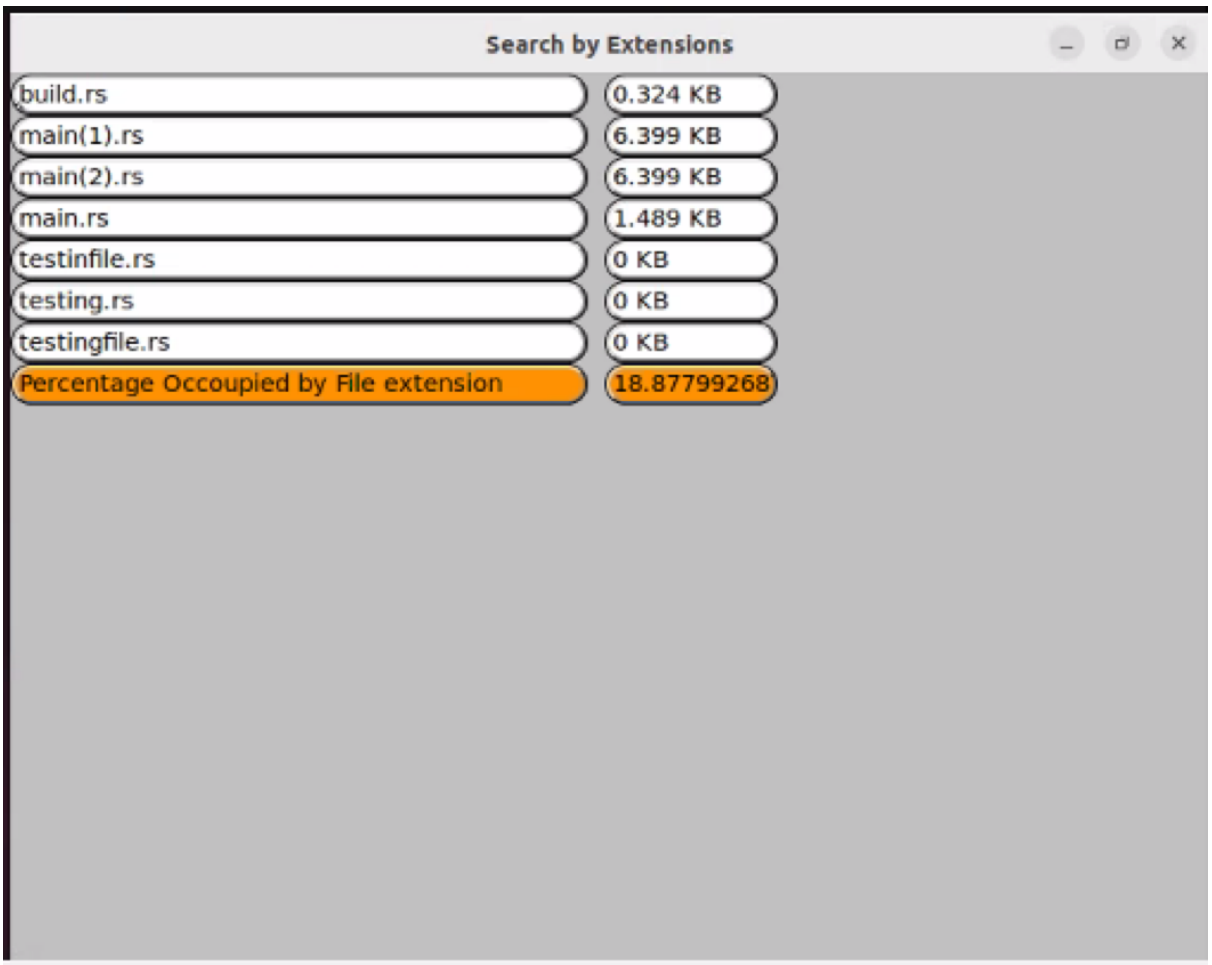
Tree form and blocks view of files and folders



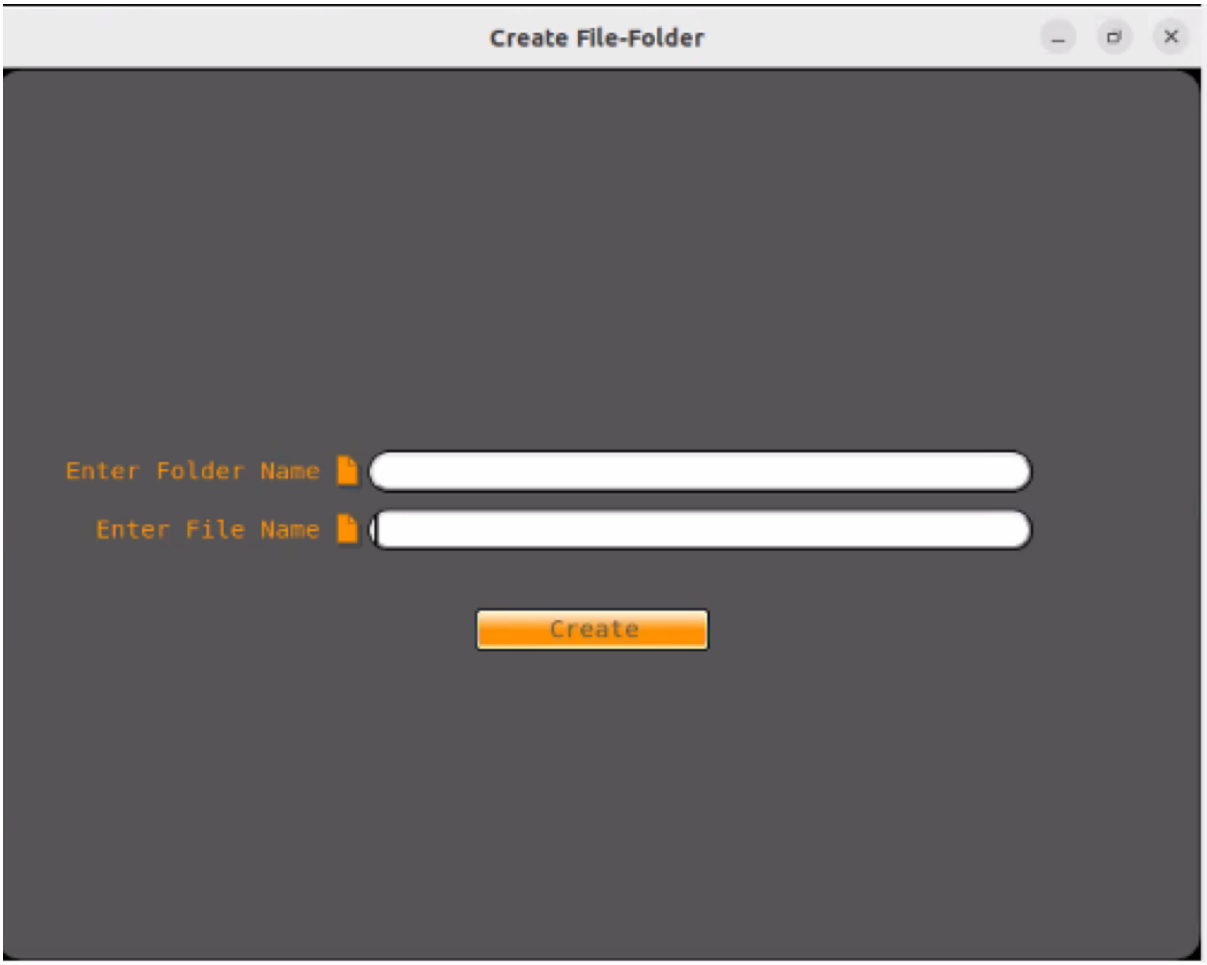
Rest of the files and folder (Scroll down)



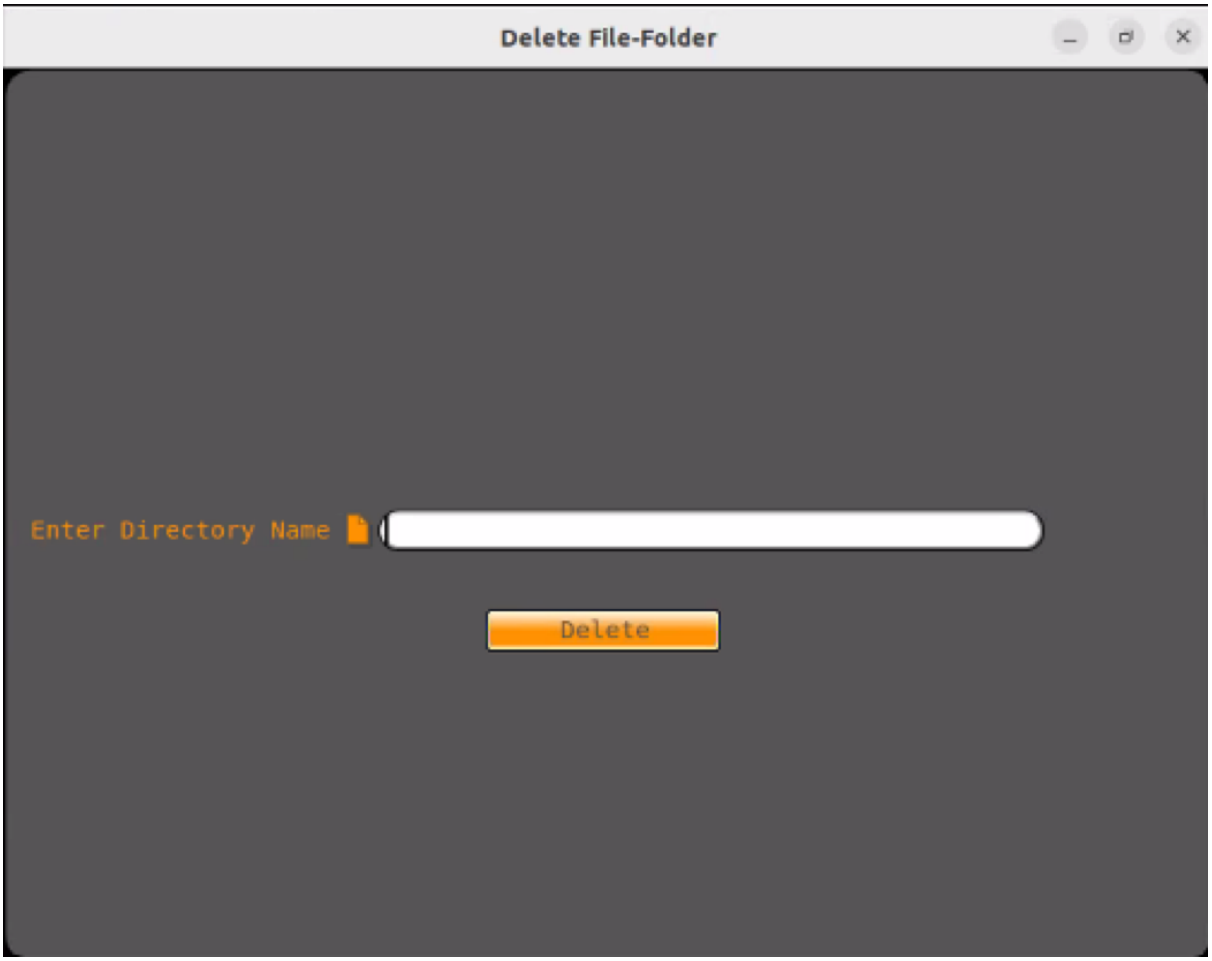
Search files by extension



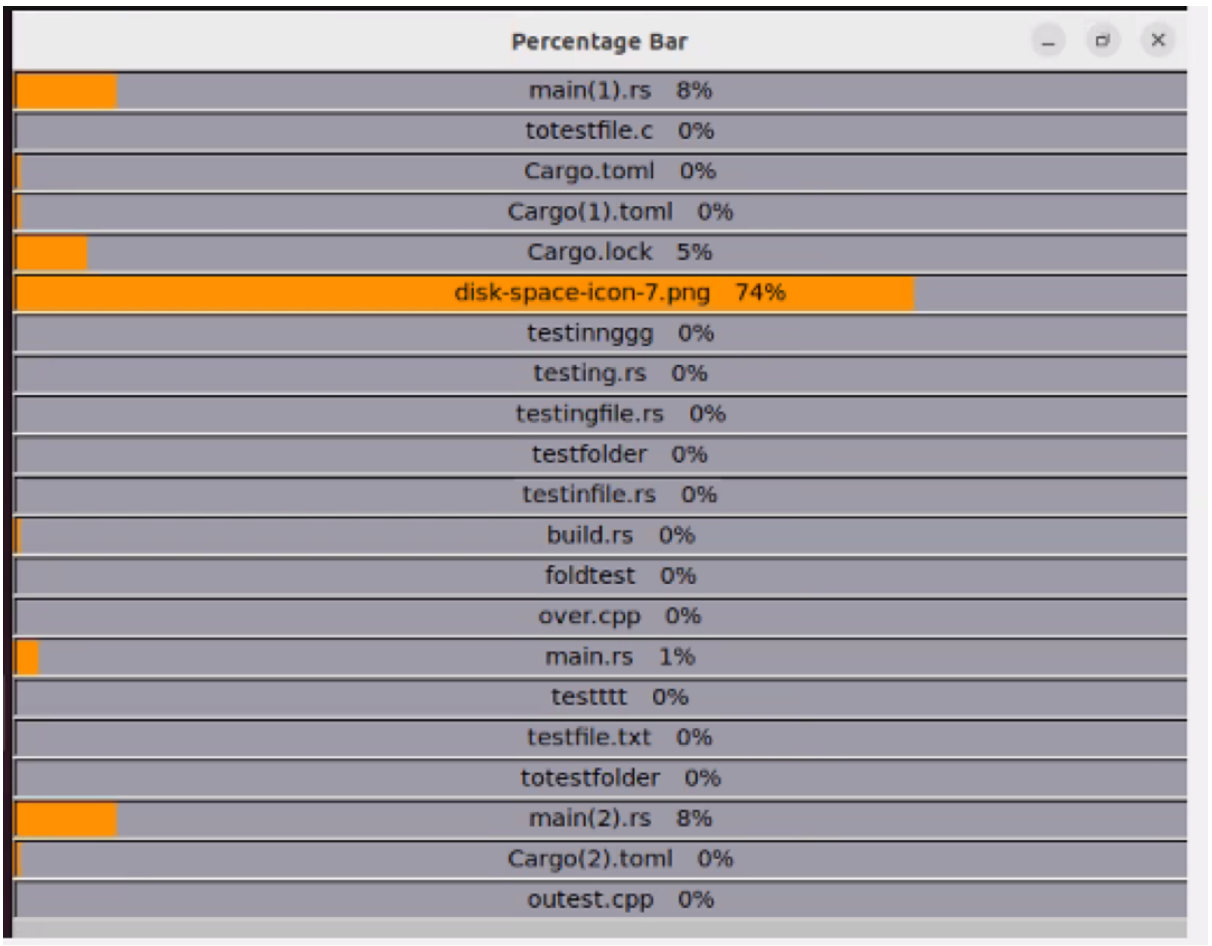
Results search files by extension



Creating file or folder



Deleting Directory



Show files sizes in progress bars showing percentage of usage from disk space

Conclusion

Concluding the above , we implemented an effective disk analyser with a unique GUI and experience, granting the user very sufficient functionalities and features. We implemented our disk analyzer fully in Rust programming language and integrated the GUI with the help of FLTK and Fluid interface libraries.