Mehran University of Engineering and Technology, Jamshoro

Department of Computer System Engineering(4th semester, 2nd year)

PROJECT REPORT: FILE SEARCH ENGINE

Name: Zameer Abbas

Roll no: 22CS0078

Subject: Operating Systems (OS) CS-261

Submitted to: Dr. Bushra Naz

Department of Computer Systems Engineering

Course: Operating Systems (CS-261)			
Instructor	Dr. Bushra Naz	Assignment Type	Complex Engineering Problem
Semester	4th	Year	2nd
Submission Deadline	5-11-2024	Assessment Score	05

Semester project is designed in a way to able students to solve the complex engineering problem using the Operating systems. Following characteristics of complex engineering problem are targeted in this semester project of OS.

Complex Engineering Problem – Characteristics		
1	Depth of knowledge Required	☑
2	Range of Conflicting Requirements	☑
3	Depth of Analysis Required	☑
4	Infrequently Encountered Issues Involved	☐
5	Beyond codes/standards of practice	☐
6	Diverse groups of stakeholders with widely varying needs involved	☐
7	Interdependence (high level problems including many components parts/sub-problems)	☐
8	Have significant consequences in a range of contexts	☐
9	Judgement (Require judgement in decision making)	☐

Project Objectives:

Rubrics				CEP characteristics	Marks distribution
	Unacceptable 2	Acceptable 8	Proficient 10		
R1: Idea/Initial Study	☐	☐	☐	WP2	20%
R2: Project Proposal	☐	☐	☐	WP1, WP3	20%
R3: Project Progress	☐	☐	☐	WP3, WP2	20%
R4: Final Report	☐	☐	☐	WP3, WP1	40%

Introduction:

The File Search Engine project, developed in Java, aims to provide a user-friendly interface for searching files on a system based on user input. It allows the user to search for specific files by name, extension, or type and open them directly from the search results. The project includes additional features such as shutdown and restart options for the system, making it versatile and interactive.

Key Features:

- **File Search:** Search for files within the specified directory or all directories by entering the file name or partial name. The system searches recursively through all folders and subfolders.
- **Open Files:** After finding the file, the user can open it directly from the search result.
- **Power Management:** The interface allows the user to shutdown or restart the computer with a simple confirmation dialog.

Programming Language:

The project is developed using Java, utilizing Java's built-in AWT (Abstract Window Toolkit) and Swing libraries to create a graphical user interface (GUI).

User Interface:

The graphical interface is intuitive, featuring:

- Text Fields for user input.
- Buttons for initiating file search, opening files, shutdown, and restart commands.
- Lists and Scroll Panes to display search results.

The interface is styled with fonts and custom images for enhanced user experience.

Project Architecture:

1. Main Class:

- Initializes the application window and GUI components.
- Handles user actions such as searching files, opening files, and system power options.

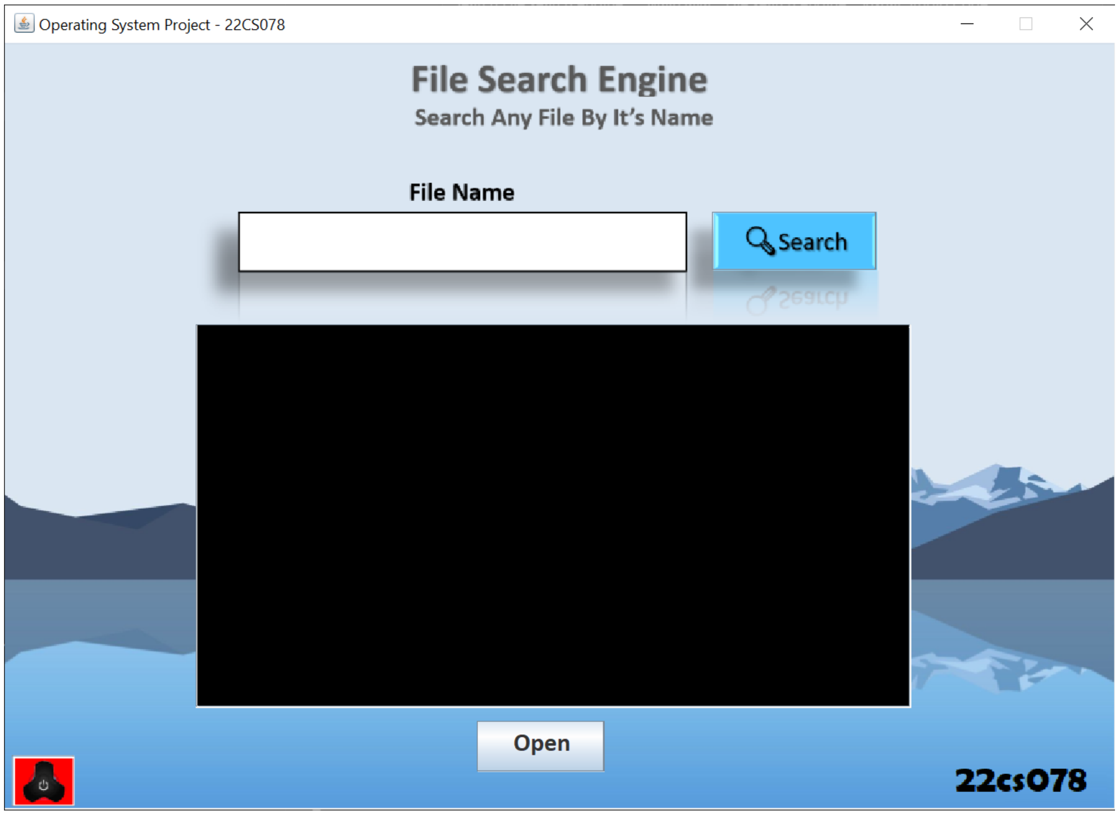
2. Search Engine:

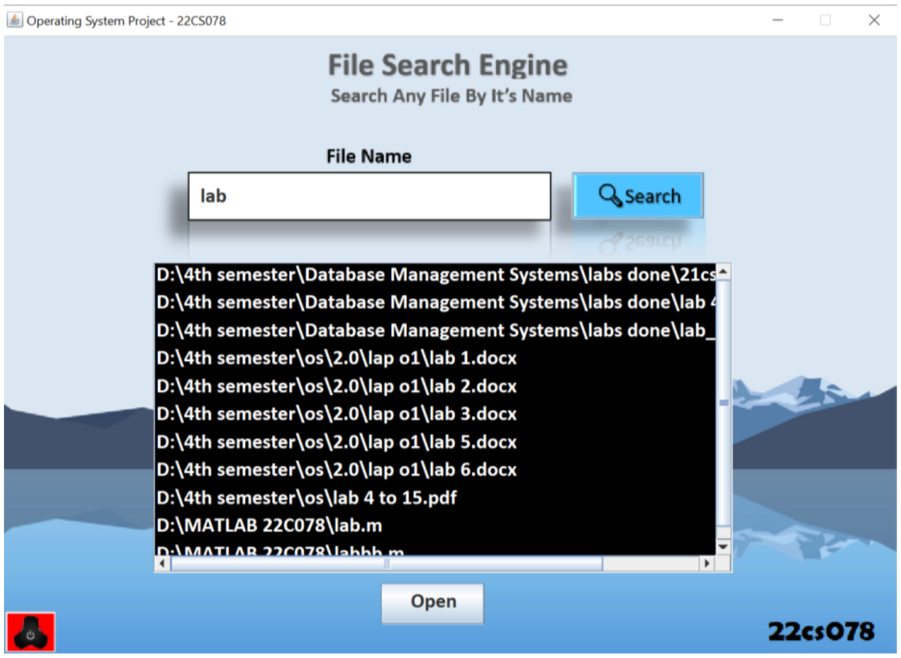
- Implements recursive file search using the Java File class.
- Supports both directory search.

3. Power Options:

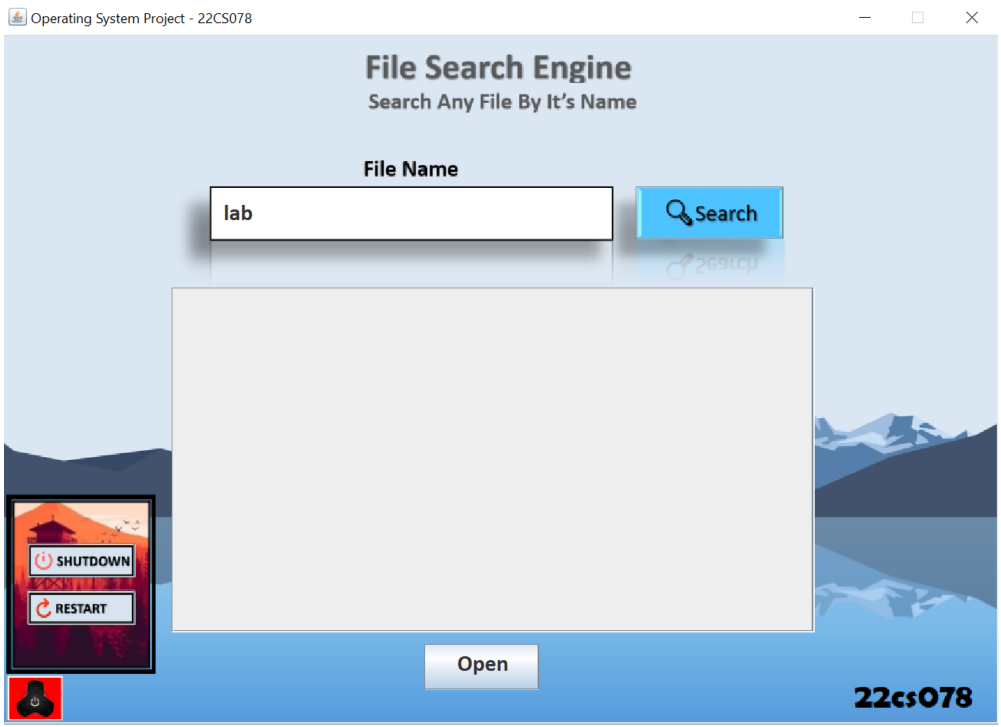
- Shuts down or restarts the system using Runtime.getRuntime().exec() to call system commands.

User interface:


Operating System Project - 22CS078
File Search Engine
Search Any File By It's Name
File Name
Search
Open
22cs078


Operating System Project - 22CS078
File Search Engine
Search Any File By It's Name
File Name
lab
Search
Open
22cs078

```
D:\4th semester\Database Management Systems\labs done\21cs
D:\4th semester\Database Management Systems\labs done\lab 4
D:\4th semester\Database Management Systems\labs done\lab_
D:\4th semester\os\2.0\lap o1\lab 1.docx
D:\4th semester\os\2.0\lap o1\lab 2.docx
D:\4th semester\os\2.0\lap o1\lab 3.docx
D:\4th semester\os\2.0\lap o1\lab 5.docx
D:\4th semester\os\2.0\lap o1\lab 6.docx
D:\4th semester\os\lab 4 to 15.pdf
D:\MATLAB 22C078\lab.m
D:\MATLAB 22C078\labbb.m
```


Operating System Project - 22CS078
File Search Engine
Search Any File By It's Name
File Name
lab
Search
SHUTDOWN
RESTART
Open
22cs078

Code Structure:

The core logic of the program is divided into methods:

- createUI(): Creates and arranges the GUI components.

```
    public void createUI() {

        t1 = new JTextField();
        t1.setBounds(195, 148, 330, 20);
        t1.setBorder(javax.swing.BorderFactory.createEmptyBorder());
        t1.setToolTipText("Enter File Name");
        t1.setText(FileName);
        t1.setFont(f);
```

```java
        c.add(t1);


        b1 = new JButton();
        b1.setBounds(554, 132, 129, 46);
        b1.setContentAreaFilled(false);
        b1.setToolTipText("Search File");
        b1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                listModel.clear();
                CallSearchEngine();
            }
        });
        c.add(b1);


        jl = new JList(listModel);
        jl.setBackground(Color.black);
        jl.setFont(f2);
        jl.setToolTipText("Search Result");
        jl.setForeground(Color.white);
        jl.addListSelectionListener((ListSelectionListener) new ListSelectionListener() {
              public void valueChanged(ListSelectionEvent evt) {
                if (!evt.getValueIsAdjusting()) {
                    if(jl.getSelectedIndex()>=0) {
                    OpenFile = jl.getSelectedValue().toString();
                    System.out.println("Clicked on : "+OpenFile);
                    }
                }
              }
            });

        scroll = new JScrollPane(jl);
        scroll.setBounds(150, 220, 560, 300);
        c.add(scroll);

        b2 = new JButton("Open");
        b2.setBounds(370, 530, 100, 40);
        b2.setFont(f);
        b2.setToolTipText("Open Selected File");
        b2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                if(ind>0) {
                    OpenFileWithCMD();
                }
            }
        });
        c.add(b2);

        ImageIcon ppbgr = new ImageIcon(getClass().getResource("pbtn1.png"));
        pwbg = new JLabel(ppbgr);
        pwbg.setBounds(6, 557, 49, 40);
        c.add(pwbg);

        shut = new JButton();
        shut.setBounds(24, 442, 94, 30);
        shut.setContentAreaFilled(false);
        shut.setToolTipText("Shutdown PC");
        shut.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                int yn = JOptionPane.showConfirmDialog(null,
                        "Shutdown Your PC? _", "Confirmation", JOptionPane.YES_NO_OPTION);

                if(yn==JOptionPane.YES_OPTION) {
                    CallPower(1);
                }
                else
                    JOptionPane.showMessageDialog(null, "Operation Cancelled -_-");
            }
        });
        c.add(shut);

        rest = new JButton();
        rest.setBounds(24, 483, 94, 30);
        rest.setContentAreaFilled(false);
        rest.setToolTipText("Restart PC");
        rest.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                int yn = JOptionPane.showConfirmDialog(null,
                        "Restart Your PC? _", "Confirmation", JOptionPane.YES_NO_OPTION);

                if(yn==JOptionPane.YES_OPTION) {
                    CallPower(2);
                }
                else
                    JOptionPane.showMessageDialog(null, "Operation Cancelled -_-");
            }
        });
        c.add(rest);

        ImageIcon mnbg = new ImageIcon(getClass().getResource("mnb.png"));
        menubg = new JLabel(mnbg);
        menubg.setBounds(6, 400, 130, 156);
        c.add(menubg);

        if(sw==1) {
            System.out.println(sw);
            menubg.setVisible(true);
            shut.setVisible(true);
            rest.setVisible(true);
            sw = 2;
        }
        else if(sw==2) {
            System.out.println(sw);
            menubg.setVisible(false);
            shut.setVisible(false);
            rest.setVisible(false);
            sw = 1;
        }

        pwr = new JButton();
        pwr.setBounds(6, 557, 49, 40);
        pwr.setContentAreaFilled(false);
        pwr.setToolTipText("Power Menu");
        pwr.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                c.removeAll();
                createUI();
            }
        });
        c.add(pwr);

        ImageIcon bgr = new ImageIcon(getClass().getResource("mainscrn.png"));
        JLabel page = new JLabel(bgr);
        page.setLayout(null);
        page.setBounds(-16, -7, 900, 610);
        page.setOpaque(true);
        c.add(page);

        repaint();
    }
```

- CallSearchEngine(): Initiates the search based on user input.

```
    public void CallSearchEngine() {

        listModel.clear();
        ind = 0;
        FileName = t1.getText();

        System.out.println("Input String is : "+FileName);

        if(FileName.isEmpty())
            JOptionPane.showMessageDialog(null, "Please Enter File Name! -_-'");
        else {
            FileNameExt();
        }
    }
    ////////////////////////////////////////////////
    public void FileNameExt() {
        int len = FileName.length();
        int i;
        boolean extTrue = false;

        for(i=len-1; i>0; i--) {
            if(FileName.charAt(i)=='.') {
                extTrue = true;
                break;
            }
        }

        if(extTrue) {
            FileSearchinit();
        }
        else {
            SearchAllFileType();
        }
    }
```

- FileSearchinit(): Performs the file search.

```
public void FileSearchinit() {

            File maindir = new File(maindirpath);
            str2[0] = maindirpath;

            srchstr = FileName;

            if(maindir.exists() && maindir.isDirectory())
            {
                File arr[] = maindir.listFiles();

                searchFile(arr,0, srchstr);
            }

            if(ind<=0) {
                System.out.println("!!!!!No Such File Found!!!!!");
                listModel.addElement("!!!!!No Such File Found!!!!!");
            }


    }
```

- searchAllFile(): Recursively searches for files across all directories.

```
    public void searchAllFile(File[] arr, int level, String srch) {
        for (File f : arr)
        {
            if(!(f.getAbsolutePath().equals(maindirpath+"System Volume Information") ||
f.getAbsolutePath().equals(maindirpath+"$RECYCLE.BIN"))) {
                if(f.isDirectory()) {
                    searchAllFile(f.listFiles(), level + 1, srch);
                }
                else if(f.isFile()) {
                    if(f.getName().contains(srch)) {
                        str2[ind] = f.getAbsolutePath();
                        System.out.println(str2[ind]);
                        listModel.addElement(str2[ind]);
                        ind++;
                    }
                }
            }
        }
    }
    public void SearchAllFileType() {

        File maindir = new File(maindirpath);
        str2[0] = maindirpath;

        srchstr = FileName;

        if(maindir.exists() && maindir.isDirectory())
        {
            File arr[] = maindir.listFiles();

            searchAllFile(arr,0, srchstr);
        }
        if(ind<=0) {
            System.out.println("!!!!!No Such File Found!!!!!");
            listModel.addElement("!!!!!No Such File Found!!!!!");
        }
    }
    public void searchFile(File[] arr, int level, String srch)
    {
        for (File f : arr)
        {

            if(!(f.getAbsolutePath().equals(maindirpath+"System Volume Information") ||
f.getAbsolutePath().equals(maindirpath+"$RECYCLE.BIN"))) {
                if(f.isDirectory()) {
                    searchFile(f.listFiles(), level + 1, srch);
                }

                else if(f.isFile()) {
                    if(srch.equals(f.getName())) {
                        str2[ind] = f.getAbsolutePath();
                        System.out.println(str2[ind]);
                        listModel.addElement(str2[ind]);
                        ind++;
                    }
                }
            }
        }
   }
```

•

- OpenFileWithCMD(): Opens the selected file using the system's default file viewer.

```java
    public void OpenFileWithCMD() {
        try {
            Desktop.getDesktop().open(new File(OpenFile));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

- CallPower(): Handles system shutdown or restart.

```java
public void CallPower(int a) {

        int x = 0;

        Runtime runtime = Runtime.getRuntime();

        if(a==1) {
            try {
                runtime.exec("shutdown -s -t " +x);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        else {
            try {
                runtime.exec("shutdown -r -t " +x);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
```

Conclusion:

The File Search Engine is an efficient tool for navigating and managing files on a computer. With its additional system control features, it serves as a convenient utility for users who need a simple yet effective way to search for and manage files. The project demonstrates the practical application of Java GUI development and system interaction using Java's built-in capabilities.