VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANASANGAMA, BELAGAVI – 590018

Mini Project Report

On

Text Editor

Submitted in partial fulfilment for the award of degree of

Bachelor of Engineering
In
Computer Science and Engineering.

Submitted by

Karthik K.K

1BG19CS043

B.N.M. Institute of Technology

An Autonomous Institution under VTU

Approved by AICTE, Affiliated to VTU, Accredited as grade A Institution by NAAC.
All UG branches – CSE, ECE, EEE, ISE & Mech.E accredited by NBA for academic years 2018-19 to 2020-21 & valid upto 30.06.2021

Post box no. 7087, 27th cross, 12th Main, Banashankari 2nd Stage, Bengaluru- 560070, INDIA
Ph: 91-80- 26711780/81/82 Email: principal@bnmit.in, www. bnmit.org

Department of Computer Science and Engineering
2021 – 2022

B.N.M. Institute of Technology

An Autonomous Institution under VTU,
Approved by AICTE, Accredited as Grade A Institution by NAAC.
All eligible UG branches – CSE, ECE, EEE, ISE & Mech. Engg. are Accredited
by NBA for academic years 2018-19 to 2021-22 & valid upto 30.06.2022
Post box no. 7087, 27th cross, 12th Main, Banashankari 2nd Stage, Bengaluru-560070, INDIA
Ph: 91-80-26711780/81/82 Email: principal@bnmit.in, www.bnmit.org

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Vidyayāmruthamashnuthe

CERTIFICATE

Certified that the mini project work entitled **Text Editor** carried out by **Karthik K.K USN 1BG19CS043,** a bonafide student of VI Semester, BNM Institute of Technology in partial fulfillment for the award of Bachelor of Engineering in COMPUTER SCIENCE AND ENGINEERING of Visvesvaraya Technological University, Belagavi during the year 2021-22. It is certified that all corrections / suggestions indicated for Internal Assessment have been incorporated in the report. The report has been approved as it satisfies the academic requirements in respect of mini project work prescribed for the said degree.

Mrs. Priyanka Padki
Assistant Professor
Department of CSE
BNMIT, Bengaluru

Dr. Chayadevi M L
Professor and HOD
Department of CSE
BNMIT, Bengaluru

ABSTRACT

Editors or text editors are software programs that enable the user to create and edit text files. In the field of programming, the term editor usually refers to source code editors that include many special features for writing and editing code. Notepad and Wordpad are some of the common editors used on Windows OS and vi, emacs, Jed, pico are the editors on UNIX OS. Features normally associated with text editors are — moving the cursor, deleting, replacing, pasting, finding, finding and replacing, saving etc. Java Swing or Swing was developed based on earlier APIs called Abstract Windows Toolkit (AWT). Swing provides richer and more sophisticated GUI components than AWT.

The GUI components are ranging from a simple label to a complex tree and table. Besides emulating the look and feel of various platforms, Swing also provides a pluggable look and feel to allow the look and feel of Java programs independent from the underlying platform. From this introduction, I am building a Java Swing based text editor with the respective functions- creating a new file, opening an old file, print the contents of a file, deleting contents, copy, cut and paste contents, change font size, etc.

ACKNOWLEDGEMENT

The completion of this project brings with a sense of satisfaction, but it is never complete without thanking the persons responsible for its successful completion.

I take this opportunity to express our profound gratitude to **Shri. Narayan Rao R Maanay**, Secretary, BNMIT, Bengaluru for his constant support and encouragement.

I would like to express my special thanks to **Prof. T. J. Rama Murthy**, Director, BNMIT, Bengaluru and **Dr. S. Y. Kulkarni**, Additional Director, BNMIT, Bengaluru for their constant guidance towards our goals and professions.

I extend my deep sense of sincere gratitude to **Dr. Krishnamurthy G. N**, Principal, BNMIT, Bengaluru, for providing us facilities required for the project.

I would also like to thank **Prof. Eshwar Maanay**, Dean , BNMIT, Bengaluru, for providing us useful suggestions required for the project.

I express my in-depth, heartfelt, sincere gratitude to **Dr. Chayadevi M L**, Professor and H.O.D, Department of Computer Science and Engineering, BNMIT, Bengaluru, for her valuable suggestions and support.

I extend my heartfelt, sincere gratitude to **Mrs. Priyanka Padki**, Assistant Professor, Department of Computer Science and Engineering, BNMIT, Bengaluru, for completion of the project.

Finally, I would like to thank all the teaching and non-teaching faculty members of Department of Computer Science and Engineering, BNMIT, Bengaluru, for their support. I would like to thank our family members and friends for their unfailing moral support and encouragement.

Karthik K.K

1BG19CS043

TABLE OF CONTENT

TABLE OF FIGURES

Chapter 1

INTRODUCTION

1.1 Overview of Text editor

A document may include some images, files, text, equations, and diagrams as well. But we will be limited to text editors only whose main elements are character strings.

The document editing process mainly compromises of the following four tasks :

The part of the document to edited or modifies is selected.

- Determining how to format this lines on view and how to display it.
- Specify and execute the operations that modify the document.
- Update the view properly.

The above steps include filtering, formatting, and traveling.

Formatting: Visibility on display screen.

Filtering: Finding out the main/important subset.

Traveling: Locating the area of interest.

User Interface of editors: The user interface of editors typically means the input, output and the interaction language. The input devices are used to enter text, data into a document or to process commands. The output devices are used to display the edited form of the document and the results of the operation/commands executed. The interaction language provides the interaction with the editor.

- **Input Devices:** Input devices are generally divided as text input, button devices and locator devices. Text device is a keyboard. Button devices are special function keys. The locator devices include the mouse. There are special voice devices as well which writes into text whatever you speak.
- **Output Devices:** TFT monitors, Printers, Teletypewriters, Cathode ray tube technology, Advanced CRT terminals.
- **Interaction language:** The interaction language could be, typing oriented or text command-oriented or could be menu oriented user interface as well. Typing or text command-oriented interaction language is very old used with the oldest editors, in the form of commands, use of functions and control keys etc.

- Menu oriented interface has a menu with the set of multiple choice of text strings. The display area is limited and the menus can be turned on/off by the user.

1.2 Editor structure

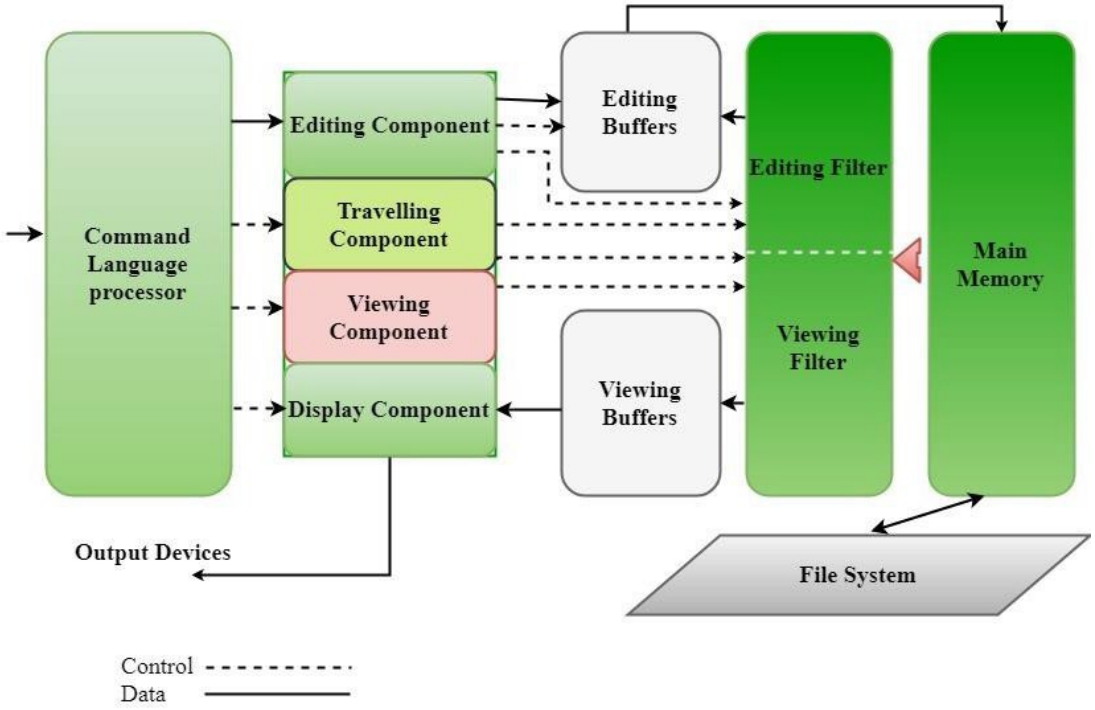

Figure 1.1: Editor structure

The command language processor accepts commands, performs functions such as editing and viewing. It involves traveling, editing, viewing and display. Editing operations are specified by the user and display operations are specified by the editor. Traveling and viewing components are invoked by the editor or the user itself during the operations.

Editing component is a module dealing with editing tasks. The current editing area is determined by the current editing pointer associated with the editing component. When editing command is made, the editing component calls the editing filter, generates a new editing buffer. Editing buffer contains the document to be edited at the current editor pointer location.

In viewing a document, the start of the area to be viewed is determined by the current viewing pointer. Viewing component is a collection of modules used to see the next view. Current viewing can be made to set or reset depending upon the last operation.

When display needs to be updated, the viewing component invokes the viewing filter, generates a new buffer and it contains the document to be viewed using the current view buffer. Then the viewing buffer is pass to the display component which produces the display by buffer mapping. The editing and viewing buffers may be identical or completely disjoint. The editing and viewing buffers can also partially overlap or can be contained one within the another. The component of the editor interacts with the document from the user on two levels: main memory and the disk files system.

1.3 Editor module's in-built functions:

We have used file reader and file writer for more information on file reading and writing in Java.
Methods used:

method	explanation
cut()	removes the selected area from the text area and store it in clipboard
copy()	copies the selected area from the text area and store it in clipboard
paste()	removes the selected area from the text area and store it in clipboard
print()	prints the components of the text area

Figure 1.2: List of in-built methods used

1.4 Editor module's user defined functions:

1. Cut: The function removes the text that's selected in the Jtextarea

```
class Cut implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            String selection = TextArea.getSelectedText();
            if (selection == null)
                return;
```

```
                StringSelection clipString = new StringSelection(selection);
                 Cboard.setContents(clipString, clipString);
                TextArea.replaceRange("", TextArea.getSelectionStart(), TextArea.getSelectionEnd());
            }
        }
```

2. Copy: The function copies the text that's selected in the Jtextarea

```
class Copy implements ActionListener {
        public void actionPerformed(ActionEvent e) {
        String selection = TextArea.getSelectedText();
        if (selection == null)
                return;
        StringSelection clipString =new StringSelection(selection);
        Cboard.setContents(clipString,clipString);
        }
}
```

3. Paste: The function displays the text that's selected in the Jtextarea

```
class Paste implements ActionListener  {
    public void actionPerformed(ActionEvent e) {
      Transferable clipData = Cboard.getContents(ClipboardJavaExample.this);
       try {
          String clipString = (String)clipData.getTransferData(DataFlavor.stringFlavor);
        TextArea.replaceRange(clipString,TextArea.getSelectionStart(),TextArea.getSelectionEnd());
       } catch(Exception ex) {
                System.err.println("Not Working");
              }
       }
}
```

1.5 Problem Statement

Write a Java program to create a simple text editor and include the following menu bars and button.

1. File menu

- open: this menu item is used to open a file
- save: this menu item is used to save a file
- print: this menu item is used to print the components of the text area
- new: this menu item is used to create a new blank file

2. Edit menu
   - cut: this menu item is to cut the selected area and copy it to clipboard
   - copy: this menu item is to copy the selected area to the clipboard
   - paste: this menu item is to paste the text from the clipboard to the text area

3. Close: this button closes the frame

Chapter 2

SYSTEM REQUIREMENTS

2.1 Software Requirements

- **Operating System:** All major operating systems (Mac, Linux, Windows) are good for java and idk installation
- **Development Platform:** Windows version 10 or 11
- **Development Tool:** Visual studio code, Eclipse
- **Language used in coding**: Java

2.2 Hardware Requirements

- Pentium or higher Processor
- 512 MB or more RAM

Chapter 3

IMPLEMENTATION

3.1 Implementation Code

Using in-built functions:

```
package com.company;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JTextArea;
import javax.swing.JScrollPane;
import javax.swing.JMenuItem;
import javax.swing.JTextField;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.JFileChooser;
import java.awt.Font;
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.*;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.io.FileReader;
import java.io.BufferedWriter;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileWriter;


class TextEditor implements ActionListener {
   JFrame f;
   JMenuBar menuBar;
   JMenu file,
         edit,
         themes,
         help;
   JTextArea textArea;
   JScrollPane scroll;
   JMenuItem darkTheme,
         moonLightTheme,
         defaultTheme,
         save,
```

```
            open,
            close,
            cut,
            copy,
            paste,
            New,
            selectAll,bold,italic,
            fontSize;
        JPanel saveFileOptionWindow;
        JLabel fileLabel, dirLabel;
        JTextField fileName, dirName;

        TextEditor(){
           f = new JFrame("Untitled_Document-1"); //setting the frame
           menuBar = new JMenuBar();

           //menus
           file = new JMenu("File");
           edit = new JMenu("Edit");
           themes = new JMenu("Themes");
           //adding menus to menu bar
           menuBar.add(file);
           menuBar.add(edit);
           menuBar.add(themes);
           f.setJMenuBar(menuBar);

           //adding submenus to file
           save = new JMenuItem("Save");
           open = new JMenuItem("Open");       //file menu
           New = new JMenuItem("New");
           close = new JMenuItem("Exit");
           file.add(open);
           file.add(New);
           file.add(save);
           file.add(close);

           cut = new JMenuItem("Cut");
           copy = new JMenuItem("Copy");        //Sub Menu for edit menu
           paste = new JMenuItem("Paste");
           selectAll = new JMenuItem("Select all");
           fontSize = new JMenuItem("Font size");
           bold = new JMenuItem("Bold");
           italic= new JMenuItem("Italic ");
           edit.add(cut);
           edit.add(copy);
           edit.add(paste);
           edit.add(selectAll);
           edit.add(fontSize);
           darkTheme = new JMenuItem("Dark Theme"); //Sub Menu for themes
           moonLightTheme = new JMenuItem("Moonlight Theme");
           defaultTheme = new JMenuItem("Default Theme");
           themes.add(darkTheme);
```

```
        themes.add(moonLightTheme);
        themes.add(defaultTheme);

        //Text area
        textArea = new JTextArea(32,88);
        f.add(textArea);

        //scroll pane
        scroll = new JScrollPane(textArea);
        scroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
        scroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
        f.add(scroll);
        cut.addActionListener(this);
        copy.addActionListener(this);
        paste.addActionListener(this);
        selectAll.addActionListener(this);
        fontSize.addActionListener(this); //change the font size
        open.addActionListener(this); //open the file
        save.addActionListener(this); //Save the file
        bold.addActionListener(this);//bold the text
        italic.addActionListener(this);//Italic the text
        New.addActionListener(this); //Create the new document
        darkTheme.addActionListener(this); //dark theme
        moonLightTheme.addActionListener(this); //moonlight theme
        defaultTheme.addActionListener(this); // default theme
        close.addActionListener(this); //close the window

        f.addWindowListener(new WindowListener() {
           @Override
           public void windowOpened(WindowEvent windowEvent) {}

           @Override
           public void windowClosing(WindowEvent e) {
              int confirmExit = JOptionPane.showConfirmDialog(f,"Do you want to exit?","Are you
    sure?.",JOptionPane.YES_NO_OPTION);

              if (confirmExit == JOptionPane.YES_OPTION)
                 f.dispose();
              else if (confirmExit == JOptionPane.NO_OPTION)
                 f.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
           }

           @Override
           public void windowClosed(WindowEvent windowEvent) {}

           @Override
           public void windowIconified(WindowEvent windowEvent) {}

           @Override
           public void windowDeiconified(WindowEvent windowEvent) {}

           @Override
```

```
            public void windowActivated(WindowEvent windowEvent) {}

            @Override
            public void windowDeactivated(WindowEvent windowEvent) {}
        });

        //Keyboard Listeners
        KeyListener k = new KeyListener() {
            @Override
            public void keyTyped(KeyEvent e) { }

            @Override
            public void keyPressed(KeyEvent e) {
                int keyCode = e.getKeyCode();
                if (keyCode == KeyEvent.VK_S && e.isControlDown())//if we click on s it will save
                    saveTheFile(); //Saving the file
            }

            @Override
            public void keyReleased(KeyEvent e) { }
        };
        textArea.addKeyListener(k);//whenever u type the below happens

        //Default Operations for frame
        f.setSize(1000,596);
        f.setResizable(true);
        f.setLocation(250,100);
        f.setLayout(new FlowLayout());
        f.setVisible(true);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        //Copy paste operations
        if (e.getSource()==cut)
            textArea.cut();
        if (e.getSource()==copy)
            textArea.copy();
        if (e.getSource()==paste)
            textArea.paste();
        if (e.getSource()==selectAll)
            textArea.selectAll();

        //change the font size by value
        if (e.getSource()==fontSize){

            String sizeOfFont = JOptionPane.showInputDialog(f,"Enter Font
Size",JOptionPane.OK_CANCEL_OPTION);
                if (sizeOfFont != null){
                    int convertSizeOfFont = Integer.parseInt(sizeOfFont);
                    Font font = new Font(Font.SANS_SERIF,Font.PLAIN,convertSizeOfFont);
```

```
                textArea.setFont(font);
            }
        }
       //Open the file
        if (e.getSource()==open){
           JFileChooser chooseFile = new JFileChooser();
           int i = chooseFile.showOpenDialog(f);
           if (i == JFileChooser.APPROVE_OPTION){//yes or no to choose a file if yes then
              File file = chooseFile.getSelectedFile(); //select the file
              String filePath = file.getPath(); //get the file path
              String fileNameToShow = file.getName(); //get the file name
              f.setTitle(fileNameToShow);

             try {
                BufferedReader readFile = new BufferedReader(new FileReader(filePath));
                String tempString1 = "";
                String tempString2 = "";

                while ((tempString1 = readFile.readLine()) != null)
                    tempString2 += tempString1 + "\n";

                textArea.setText(tempString2);
                readFile.close();
             }catch (Exception ae){
                ae.printStackTrace();
             }
           }
        }

        //Save the file
        if (e.getSource()==save) saveTheFile();


        //New menu operations
        if (e.getSource()==New) textArea.setText("");


        //Exit from the window
        if (e.getSource()==close) System.exit(1);


        //themes area
        if (e.getSource()==darkTheme){
           textArea.setBackground(Color.DARK_GRAY);       //dark Theme
           textArea.setForeground(Color.WHITE);
        }

        if (e.getSource()==moonLightTheme){
           textArea.setBackground(new Color(107, 169, 255));
           textArea.setForeground(Color.black);
        }
```

```
            if (e.getSource() == defaultTheme){
               textArea.setBackground(new Color(255, 255, 255));
               textArea.setForeground(Color.black);
            }
         }

      //Save the file
         public void saveTheFile(){
            saveFileOptionWindow = new JPanel(new GridLayout(2,1));
            fileLabel = new JLabel("Filename      :- ");
            dirLabel = new JLabel("Save File To :- ");
            fileName = new JTextField();
            dirName = new JTextField();
            saveFileOptionWindow.add(fileLabel);
            saveFileOptionWindow.add(fileName);
            saveFileOptionWindow.add(dirLabel);
            saveFileOptionWindow.add(dirName);

            JOptionPane.showMessageDialog(f,saveFileOptionWindow); //show the saving dialogue box
            String fileContent = textArea.getText();
            String file = fileName.getText();
            String filePath = dirName.getText()+"/"+file+".txt";

            try {
               BufferedWriter writeContent = new BufferedWriter(new FileWriter(filePath));
               writeContent.write(fileContent);
               writeContent.close();
               JOptionPane.showMessageDialog(f,"File Successfully saved!");
            }catch (Exception ex){
               ex.printStackTrace();
            }
         }

         public static void main(String[] args) {
            new TextEditor();
         }
      }
```

Chapter 4

RESULTS

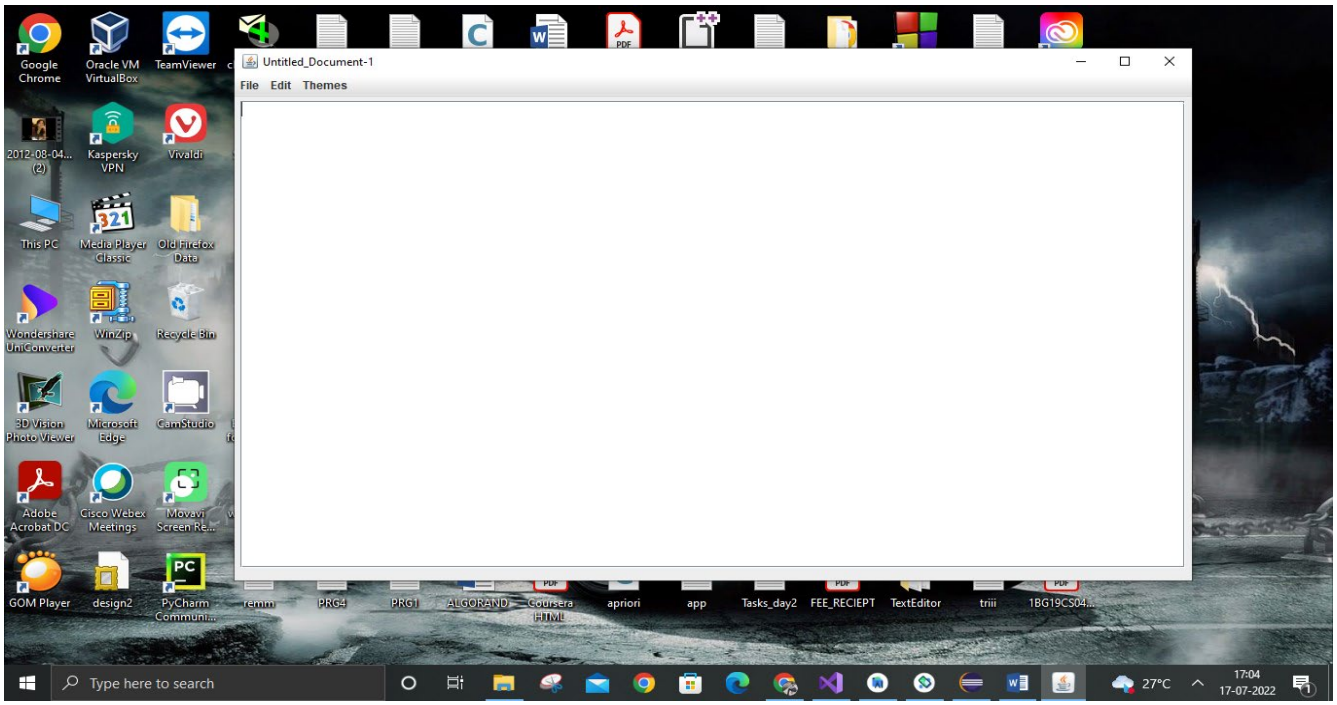

Figure 4.1: GUI for the TextEditor

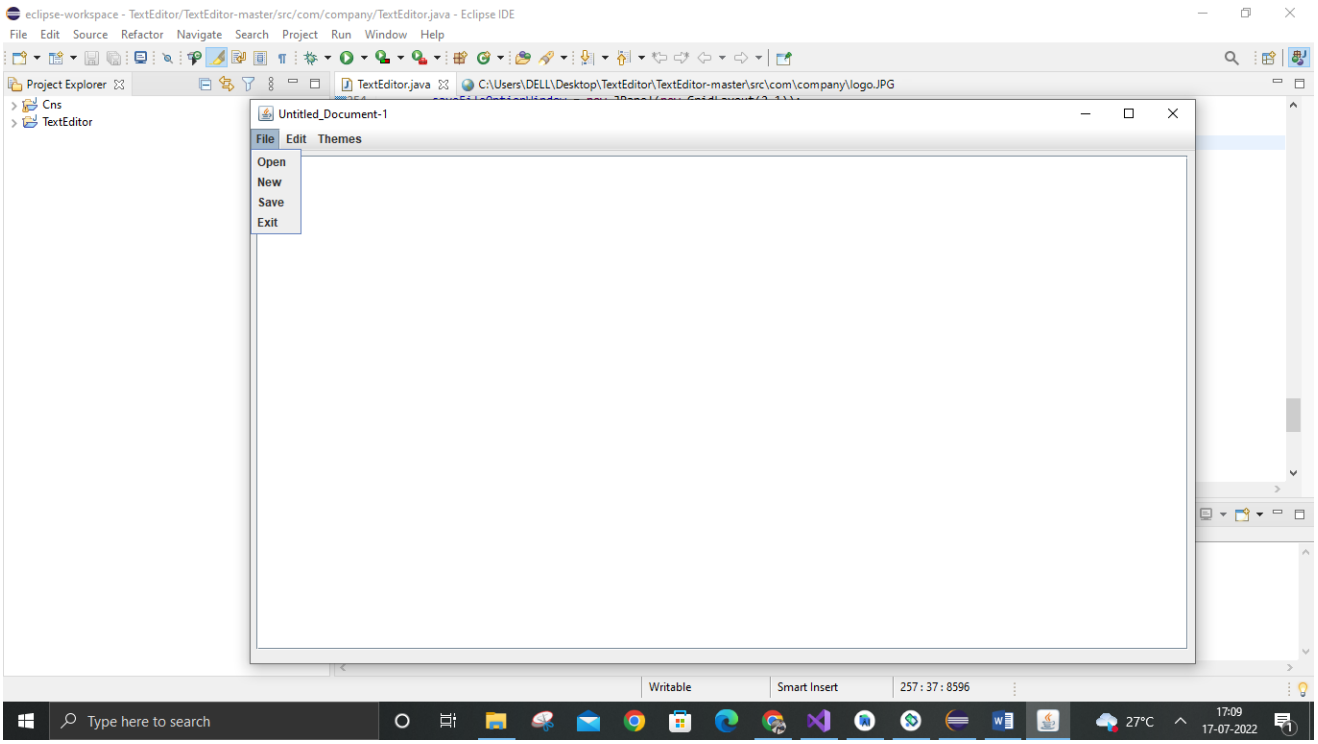

Figure 4.2 File Menu Menu

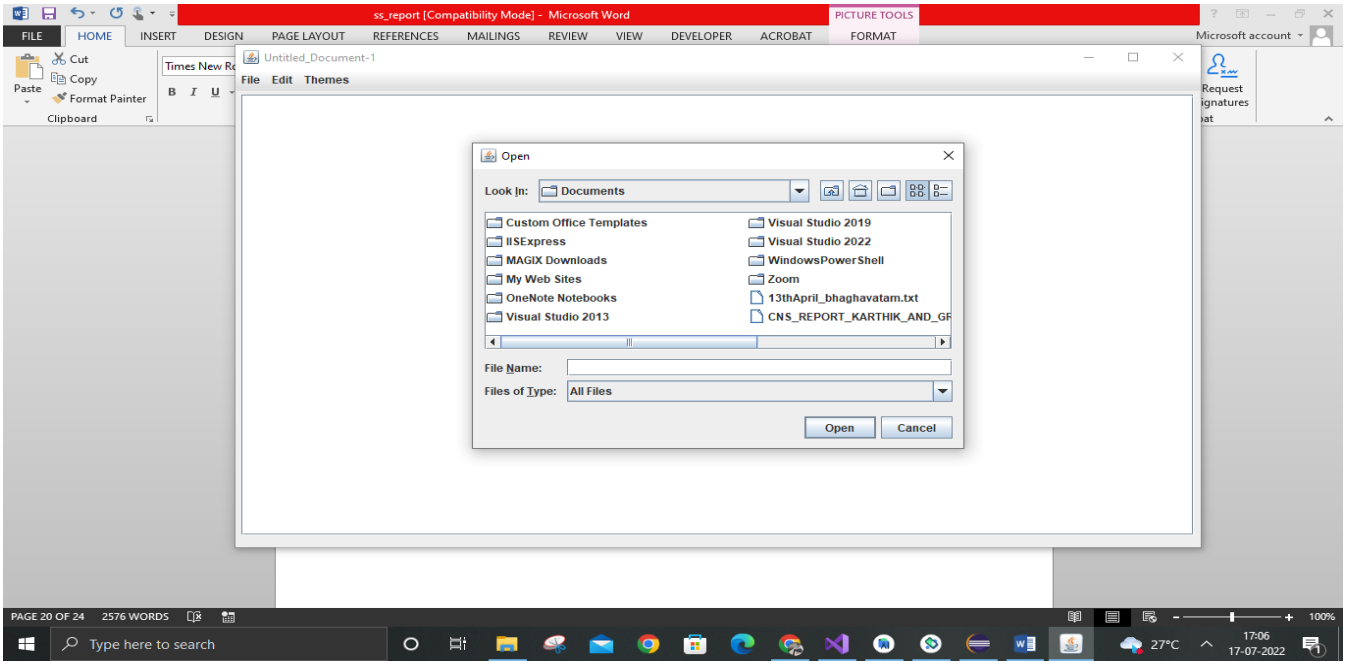

Figure 4.3: GUI for opening a file

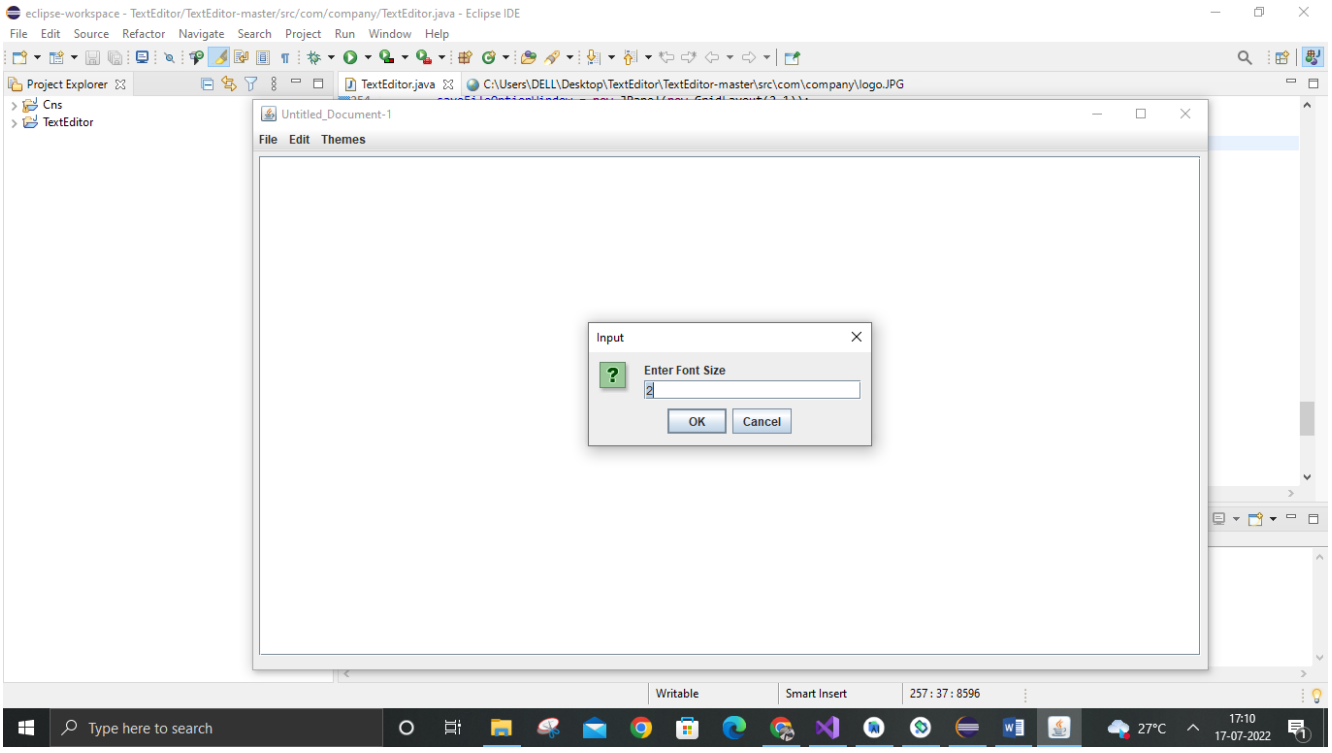

Figure 4.4 GUI for FontSize

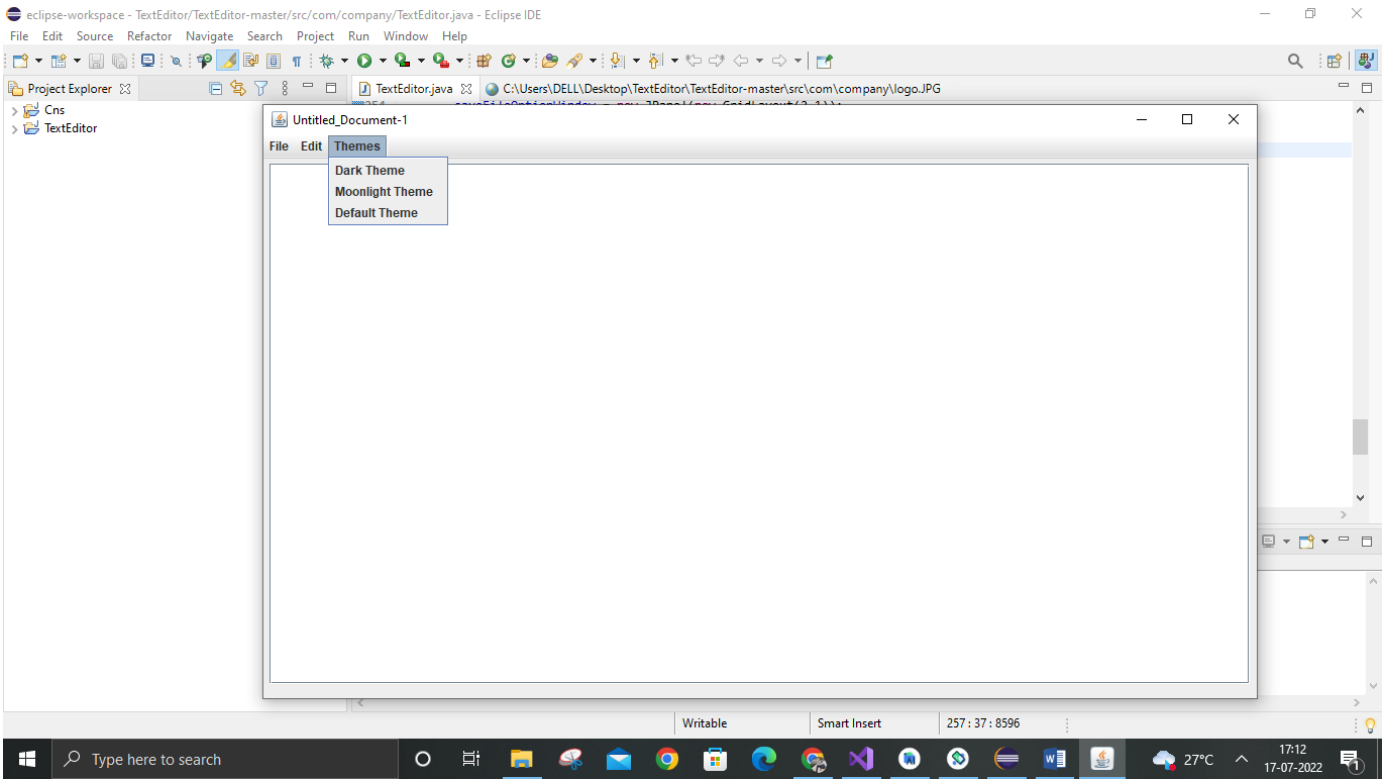

Figure 4.5: GUI for Themes menu

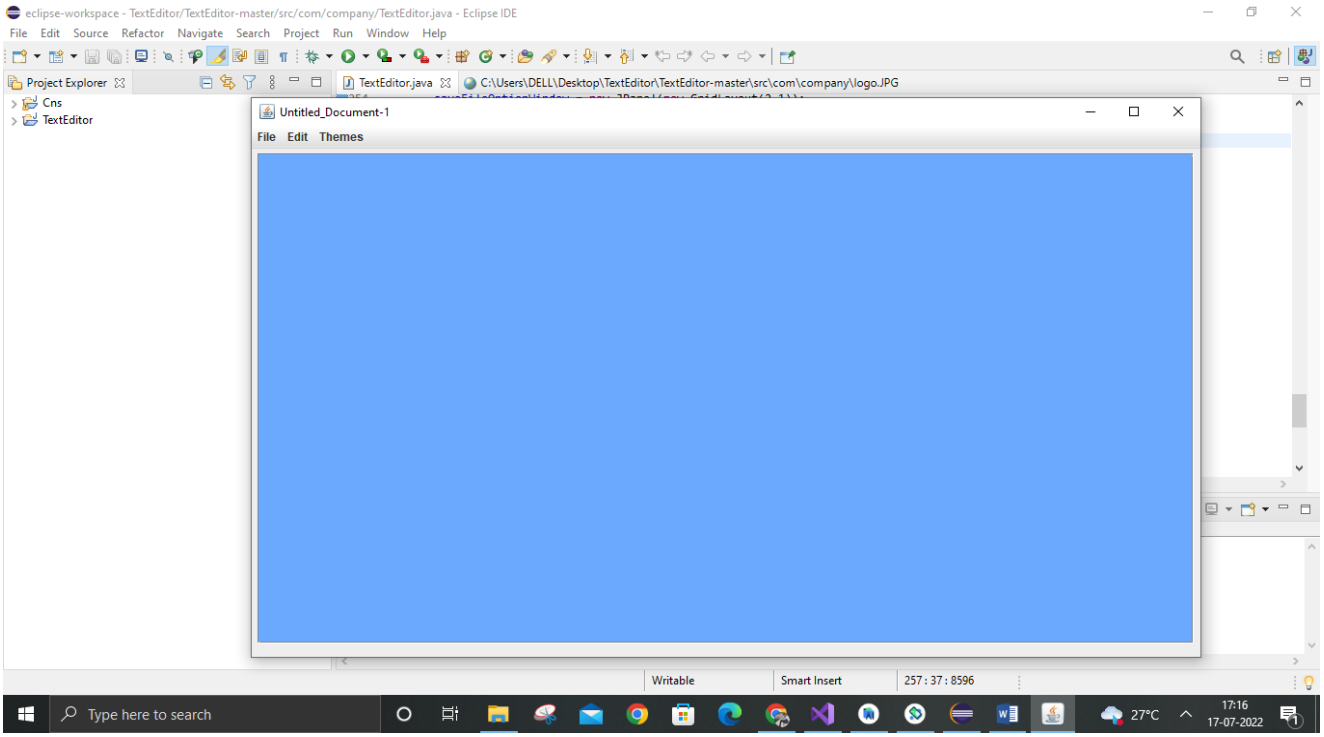

Figure 4.6: Theme changed to MoonLight

Chapter 6

CONCLUSION

6.1 Conclusion

This project uses the basic java programming concepts to materialize a user friendly GUI and thus simulate a basic text editor. Here we have shown a simple code in which one can use a set of in-built functions once those structure for each object on the GUI is defined. Functions to make the program interactive for the user have been provided along with those that allow users to experience an extensive overview of the Java Swing functions. The list of methods to read the file and write into a file has already been mentioned in the beginning of the report which make it possible to display a text editor. While rendering the GUI we can also specify the theme of the GUI. The variety of themes include system, motif, window, metal look and feel. The current theme being used is windows look and feel.

6.2 References

- System Software by Leland. L. Beck, D Manjula, 3rd edition, 2012 2.
- Alfred V Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman , Compilers-Principles, Techniques and Tools, Pearson, 2nd edition, 2007
- https://www.geeksforgeeks.org/java-swing-create-a-simple-text-editor/
- https://www.geeksforgeeks.org/editors-types-system-programming/
- https://www.zentut.com/java-swing/introduction-to-java-swing/