LOVELY PROFESSIONAL UNIVERSITY

CSE310 : PROGRAMMING IN JAVA

TITLE: JAVA PROJECT ON TO-DO LIST APP

NAME : ASHIN VINCENT
REG NO : 12105321
SECTION : K21QA
ROLL NO : 16

Roles and Responsibilities

Name: Ashin Vincent Reg no:12105321	1. Designing and developing the user interface for the application. 2. Implementing the functionality for adding, editing, and deleting tasks in the to-do list. 3. Using the Java Swing library for creating graphical components. 4. Defining and implementing the action listeners for the buttons. 5. Creating a custom font and colors for the user interface. 6. Managing the data of the to-do list using an ArrayList. 7. Creating a JList and a JScrollPane for displaying the list of tasks. 8. Testing and debugging the application to ensure its proper functioning.

CONTENTS

Abstract

The To-Do List application is a simple but useful program designed to help users keep track of their tasks and to-dos. The application allows users to add, edit, and delete tasks, as well as view their list of tasks at any time. The program was developed using Java and the Swing user interface toolkit, making it portable and easy to use on any platform.The application's interface was designed to be aesthetically pleasing and easy to use, while also being functional and efficient.

The purpose of this project report is to provide a detailed overview of the development process, features, and functionality of the To-Do List application. The report will cover the design and implementation of the application, including a flow chart and use case diagram to help illustrate the program's structure and operation.

Overall, the To-Do List application is a valuable tool for anyone looking to organize their tasks and stay on top of their to-dos. Whether used for personal or professional tasks, the application is easy to use and provides a straightforward way to manage your daily tasks and priorities.

Introduction

TO-DO LIST APPLICATION

With the growth of technology and the digital age, there is a need for a tool that can assist people in keeping track of their daily tasks and to-do lists. Many people still use traditional methods such as writing their to-do lists on a piece of paper or using a notepad, but these methods can often be unreliable and inefficient.

To address this need, the To-Do List application was developed. The To-Do List application is a software tool designed to help users keep track of their daily tasks and to-do lists. It is an intuitive and user-friendly application that provides an easy way for individuals to manage their tasks, set reminders, and prioritize their work. The To-Do List application is a useful tool for busy professionals, students, and anyone who wants to keep track of their daily tasks and ensure that they get things done in a timely and efficient manner.

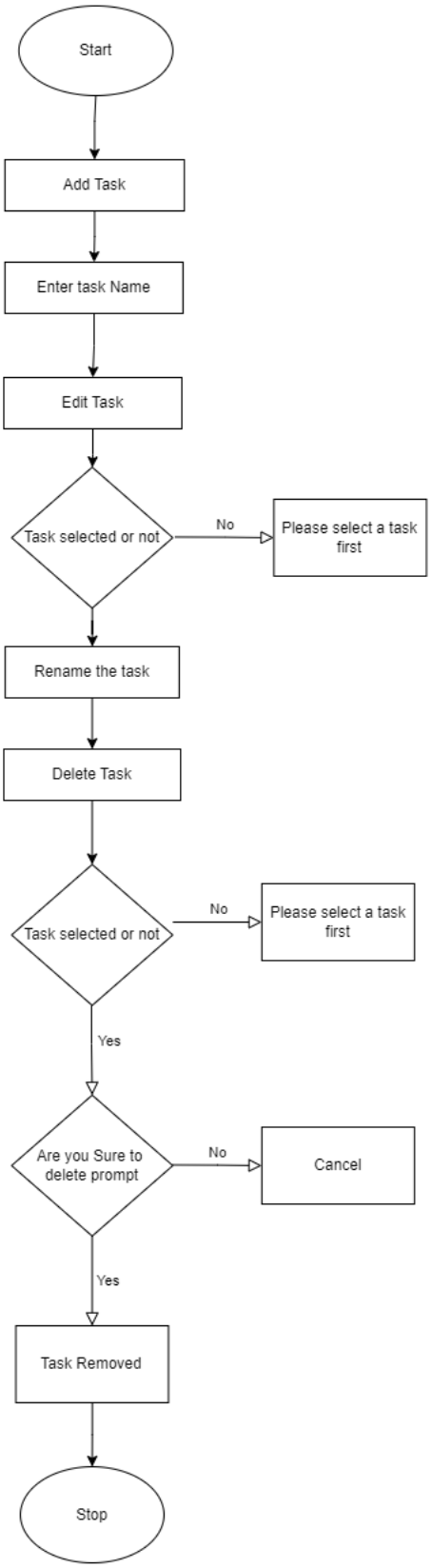
Objectives and scope of the project:

The objective of this project is to develop a simple, user-friendly, and efficient to-do list application. The application will allow users to create, edit, and delete tasks, as well as organize them into categories if desired. The goal is to

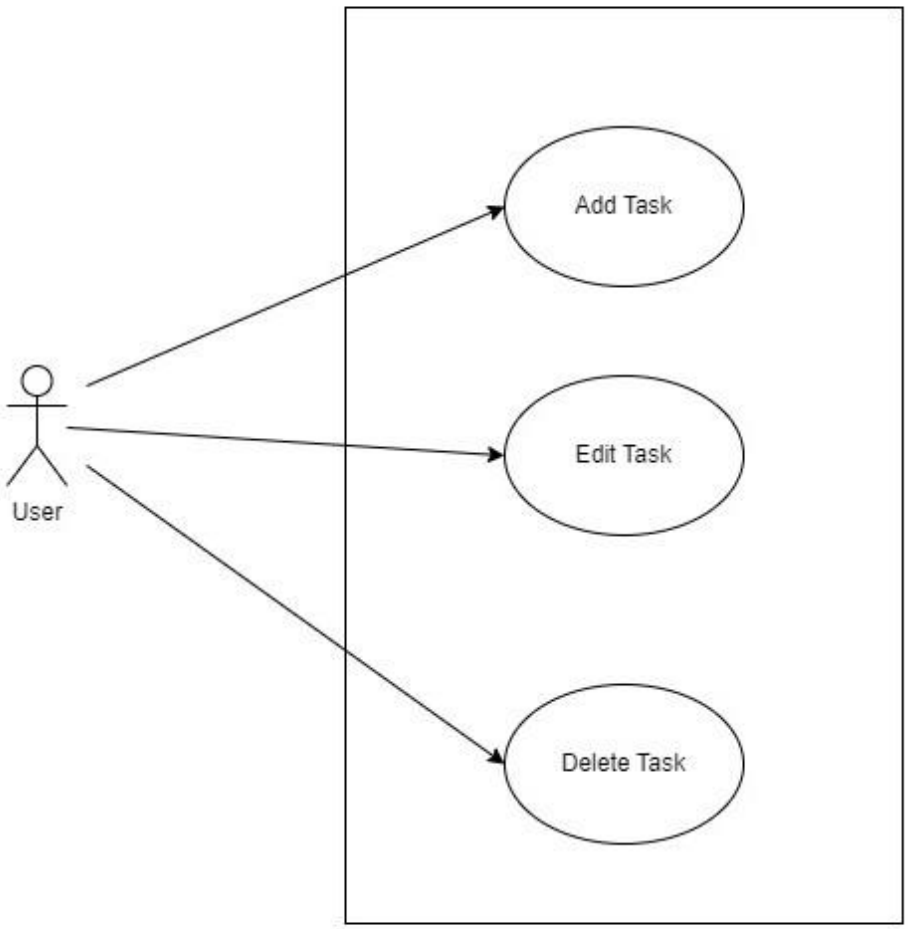
provide a solution that can help users keep track of their daily tasks, assignments, and deadlines, and improve their productivity.
The scope of the project is limited to the development of a desktop-based application using Java programming language. The application will have a graphical user interface (GUI) designed using the Swing library. The project will not involve any server-side or cloud-based functionality, as the application will run locally on the user's machine. The application will be developed in an agile manner, with frequent feedback and iterations from the stakeholders and users.

The target audience for this application is anyone who needs to manage their daily tasks and schedules, including students, professionals, and individuals who work from home. The application will be designed to be intuitive and easy to use, with clear and concise instructions for new users. It will be tested rigorously to ensure that it meets the requirements and quality standards set by the stakeholders.

FLOWCHART


Start
Add Task
Enter task Name
Edit Task
Task selected or not
No
Please select a task first
Rename the task
Delete Task
Task selected or not
No
Please select a task first
Yes
Are you Sure to delete prompt
No
Cancel
Yes
Task Removed
Stop

USE CASE DIAGRAM

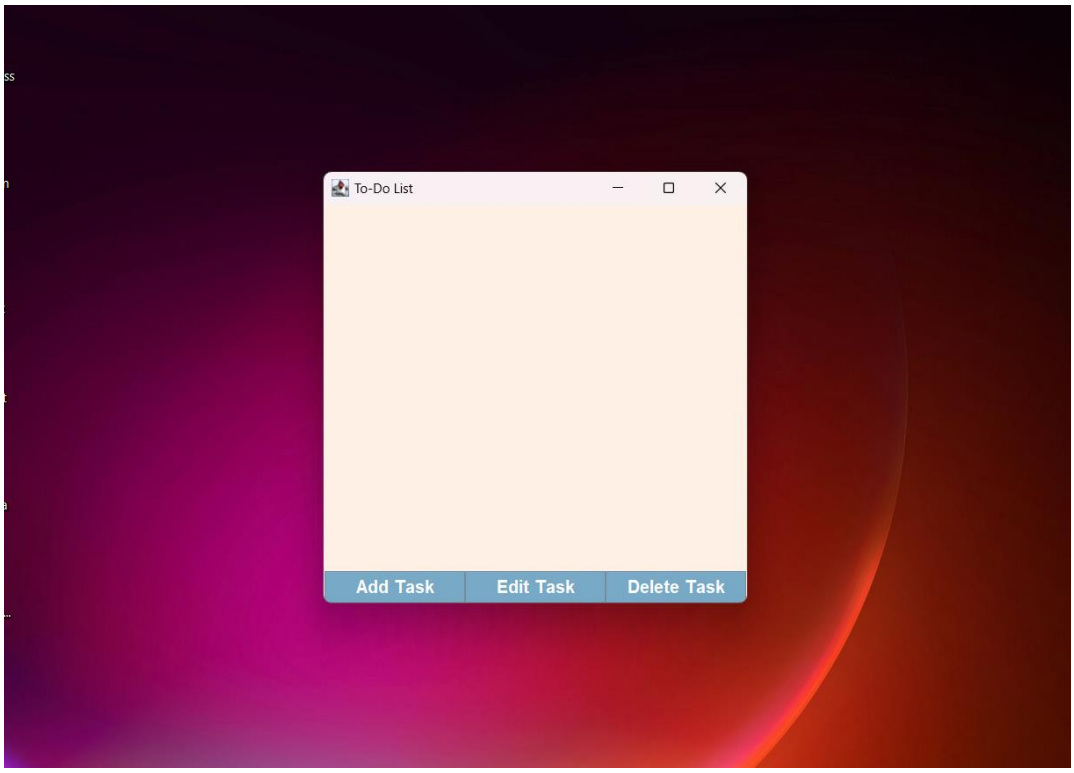


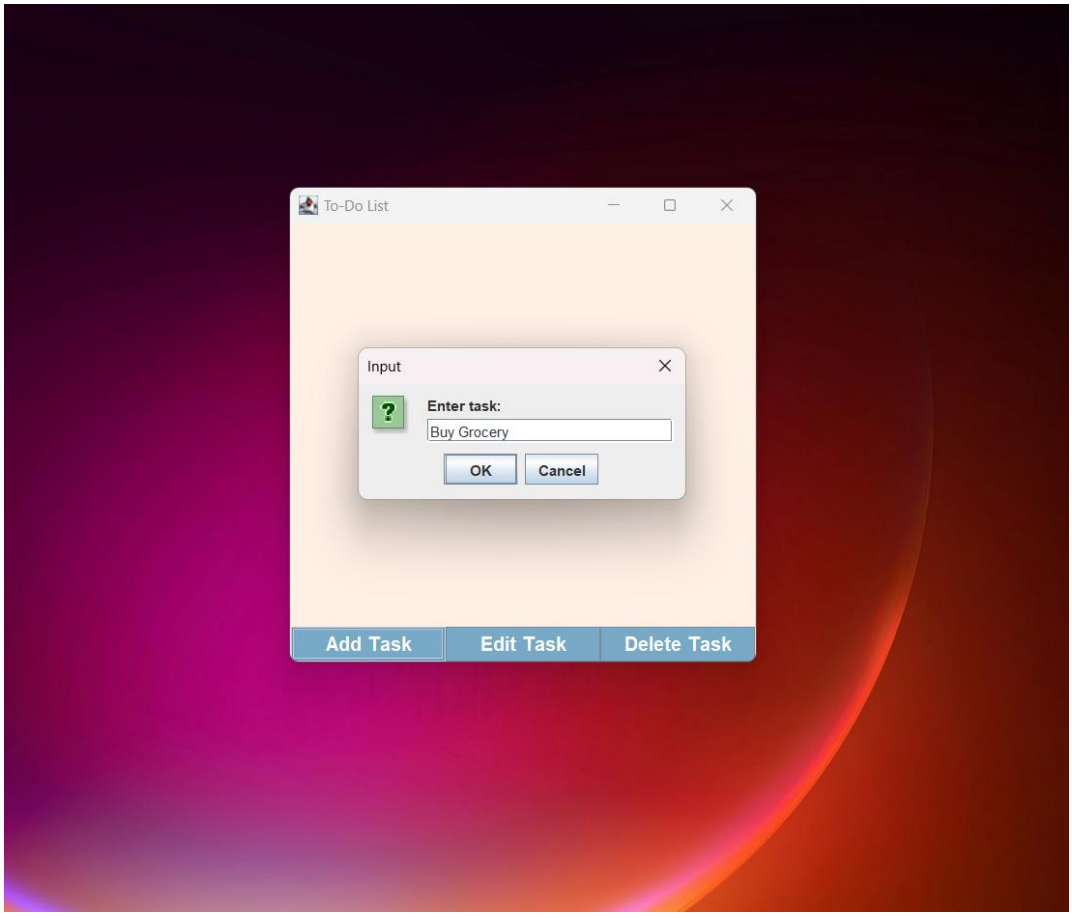
CODE

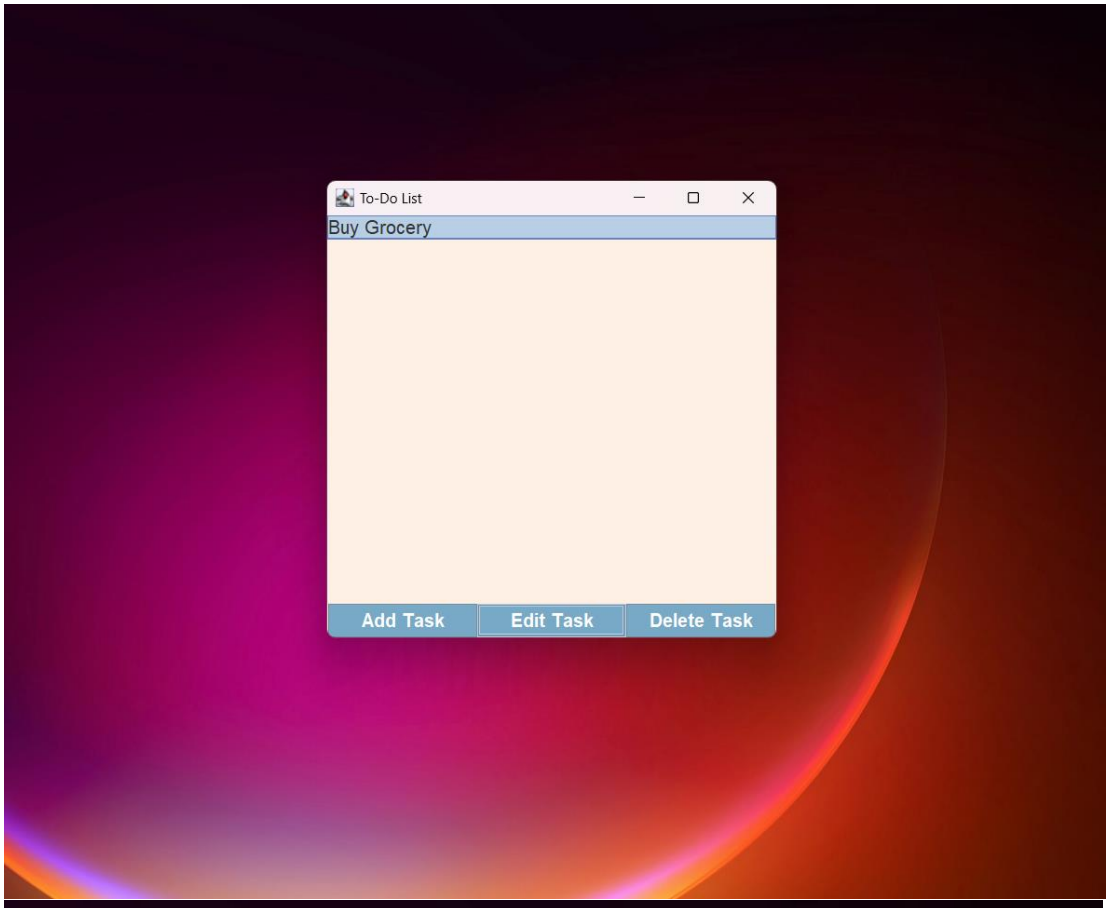
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;

public class ToDoList extends JFrame implements ActionListener {

   private final ArrayList<String> tasks;
   private final DefaultListModel<String> listModel;
   private final JList<String> taskList;
   private final JButton addButton;
   private final JButton editButton;
   private final JButton deleteButton;
   private final Color backgroundColor;
   private final Color textColor;
   private final Font font;

   public ToDoList() {
       super("To-Do List");
       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       setSize(400, 400);

       backgroundColor = new Color(255, 240, 230); // light peach color
       textColor = new Color(80, 80, 80); // dark gray color
       font = new Font("Arial", Font.PLAIN, 16); // custom font

       tasks = new ArrayList<>();
       listModel = new DefaultListModel<>();
       taskList = new JList<>(listModel);
       taskList.setBackground(backgroundColor);
       taskList.setForeground(textColor);
       taskList.setFont(font);

       JPanel buttonPanel = new JPanel(new GridLayout(1, 3));
       buttonPanel.setBackground(backgroundColor);

       addButton = new JButton("Add Task");
       addButton.addActionListener(this);
       addButton.setBackground(new Color(120, 170, 200)); // blue color
       addButton.setForeground(Color.white);
       addButton.setFont(font.deriveFont(Font.BOLD));
       buttonPanel.add(addButton);

       editButton = new JButton("Edit Task");
       editButton.addActionListener(this);
       editButton.setBackground(new Color(120, 170, 200)); // blue color
       editButton.setForeground(Color.white);
       editButton.setFont(font.deriveFont(Font.BOLD));
       buttonPanel.add(editButton);
```

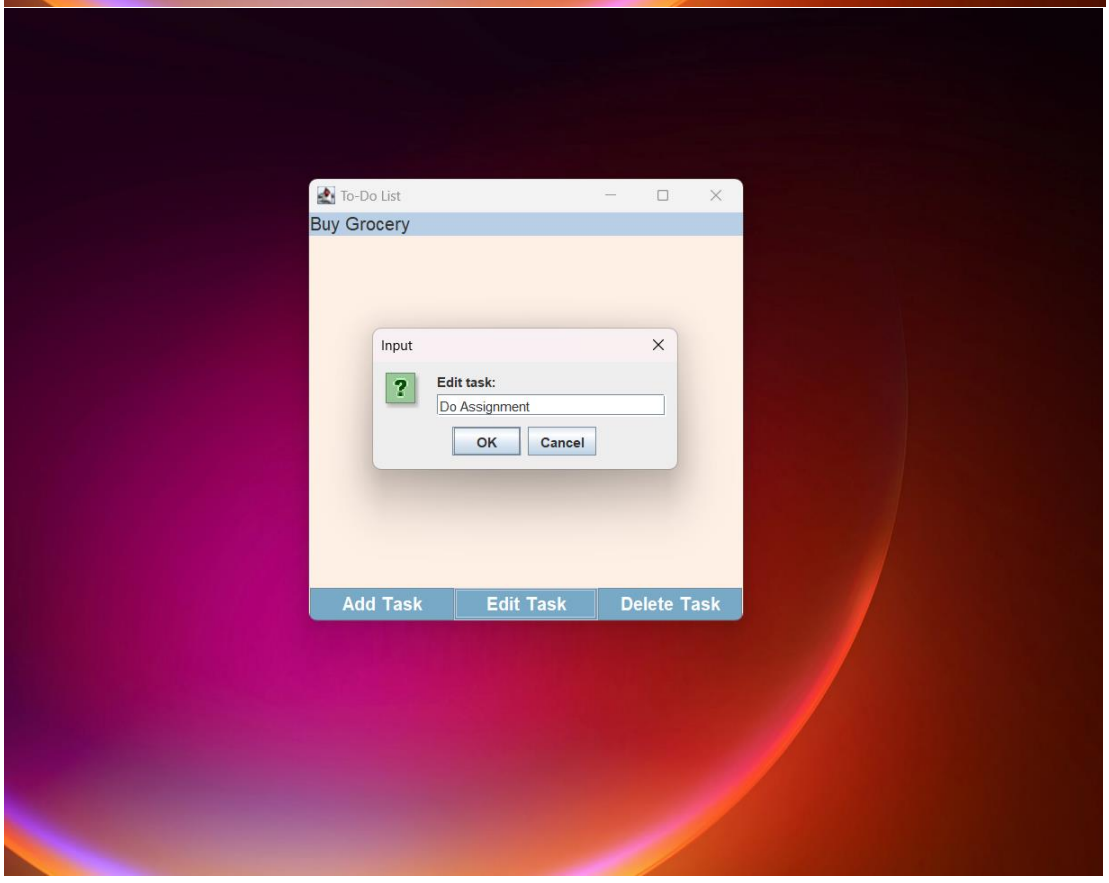
```
        deleteButton = new JButton("Delete Task");
        deleteButton.addActionListener(this);
        deleteButton.setBackground(new Color(120, 170, 200)); // blue color
        deleteButton.setForeground(Color.white);
        deleteButton.setFont(font.deriveFont(Font.BOLD));
        buttonPanel.add(deleteButton);

        JScrollPane scrollPane = new JScrollPane(taskList);
        scrollPane.setBackground(backgroundColor);
        scrollPane.setBorder(BorderFactory.createEmptyBorder());
        add(scrollPane, BorderLayout.CENTER);
        add(buttonPanel, BorderLayout.SOUTH);
        getContentPane().setBackground(backgroundColor);
        setVisible(true);
    }

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

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == addButton) {
            String task = JOptionPane.showInputDialog(this, "Enter task:");
            if (task != null && !task.isEmpty()) {
                tasks.add(task);
                listModel.addElement(task);
            }
        } else if (e.getSource() == editButton) {
            int selectedIndex = taskList.getSelectedIndex();
            if (selectedIndex == -1) {
                JOptionPane.showMessageDialog(this, "Please select a task to
edit.");
                return;
            }

            String currentTask = tasks.get(selectedIndex);
            String newTask = JOptionPane.showInputDialog(this, "Edit task:",
currentTask);
            if (newTask != null && !newTask.isEmpty()) {
                tasks.set(selectedIndex, newTask);
                listModel.set(selectedIndex, newTask);
            }
        } else if (e.getSource() == deleteButton) {
            int selectedIndex = taskList.getSelectedIndex();
            if (selectedIndex == -1) {
                JOptionPane.showMessageDialog(this, "Please select a task to
delete.");
                return;
            }

            int confirm = JOptionPane.showConfirmDialog(this, "Are you sure
you want to delete this task?");
            if (confirm == JOptionPane.YES_OPTION) {
                tasks.remove(selectedIndex);
                listModel.remove(selectedIndex);
            }
        }
    }
}
```

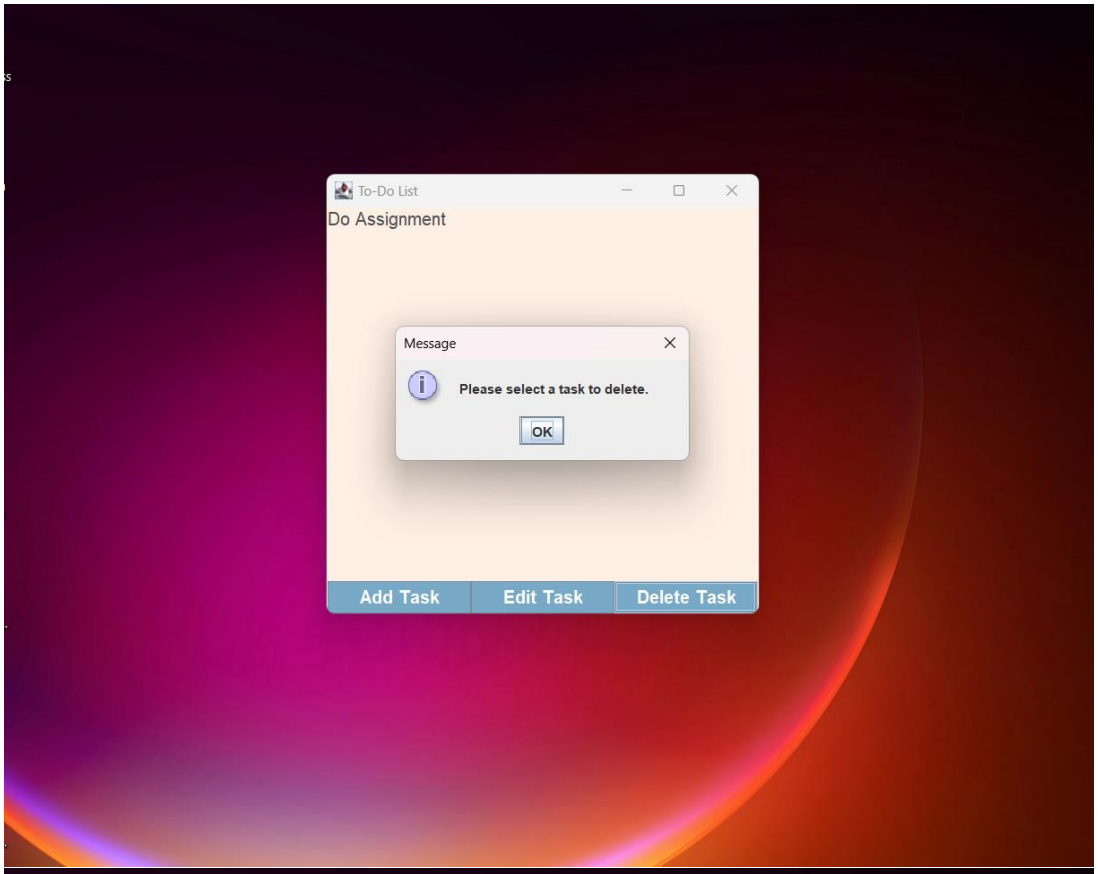
Screenshots

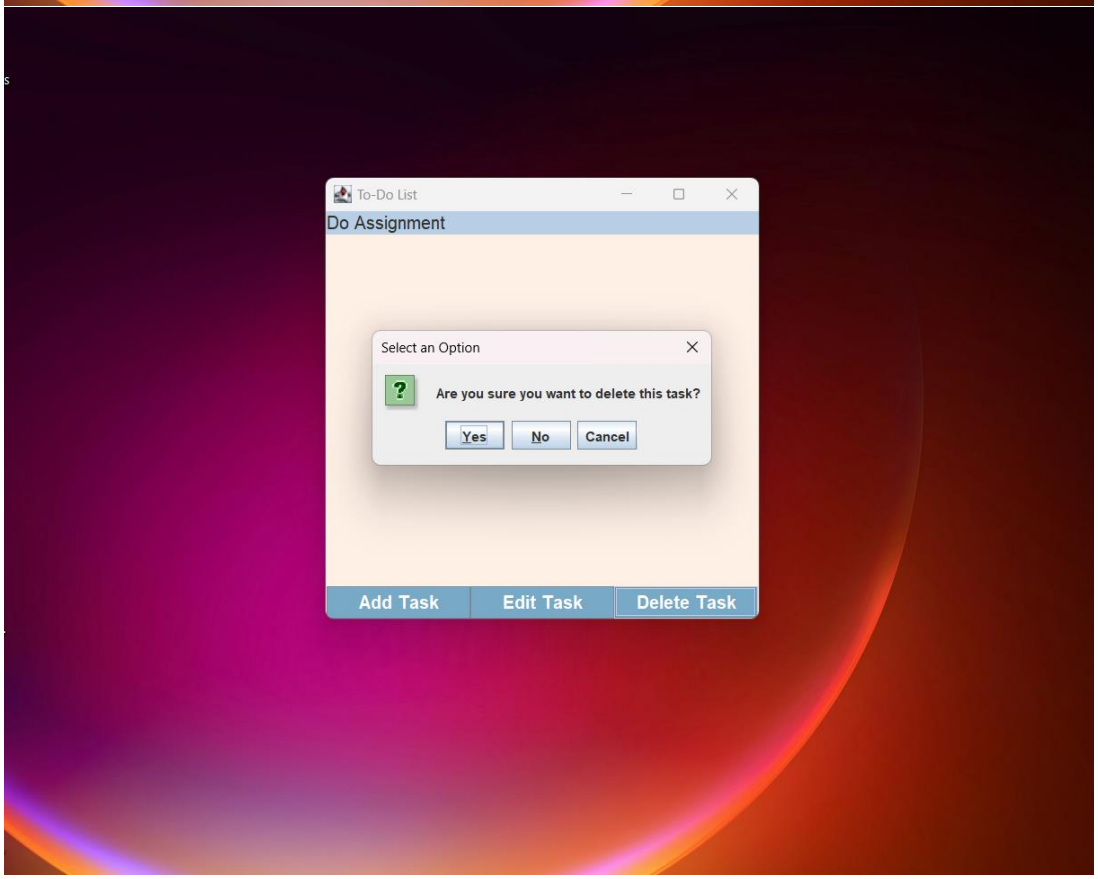





To-Do List
Buy Grocery
Add Task
Edit Task
Delete Task


To-Do List
Buy Grocery
Input
Edit task:
Do Assignment
OK
Cancel
Add Task
Edit Task
Delete Task


To-Do List
Do Assignment
Message
Please select a task to delete.
OK
Add Task
Edit Task
Delete Task


To-Do List
Do Assignment
Select an Option
Are you sure you want to delete this task?
Yes
No
Cancel
Add Task
Edit Task
Delete Task

Implementation

The implementation of the To-Do List application involved the use of Java programming language and the Swing GUI toolkit. The source code was written using an IDE such as Eclipse.

The application has a main class named ToDoList which extends the JFrame class and implements the ActionListener interface. This class contains the graphical user interface components such as buttons, list, and panels. The components were created using Swing and AWT libraries.

The ToDoList class has a constructor that initializes the components and sets their properties such as font, color, and size. It also adds event listeners to the buttons.

The actionPerformed method is implemented to handle the button clicks. When the user clicks the "Add Task" button, a dialog box is displayed asking the user to enter a new task. If the user enters a non-empty string, the task is added to the list and displayed in the UI.

When the user clicks the "Edit Task" button, the application checks if a task is selected in the list. If a task is selected, a dialog box is displayed with the selected task. The user can then edit the task and click the "OK" button to save the changes. If the user cancels the dialog box, the changes are not saved.

When the user clicks the "Delete Task" button, the application checks if a task is selected in the list. If a task is selected, a confirmation dialog box is displayed asking the user if they want to delete the task. If the user clicks the "Yes" button, the task is removed from the list and the UI is updated.

The implementation of the To-Do List application also involved the use of object-oriented programming principles such as encapsulation, inheritance, and polymorphism. The use of ArrayList and DefaultListModel classes made it easier to manage the list of tasks in the application.

Conclusion

In conclusion, the To-Do List application is a simple yet useful tool that helps users keep track of their tasks and organize their work efficiently. The application is designed to be user-friendly, with a clear and intuitive interface that allows users to add, edit, and delete tasks quickly and easily. It also provides a range of customization options, such as the ability to change the background color and font, to make the application more personalized and appealing to individual users.

The development of this application was a challenging yet rewarding process that required careful planning, design, and implementation. The use of Java and the Swing library provided a solid foundation for creating a robust and reliable application that can be used on various operating systems. The application was also tested thoroughly to ensure that it met the requirements and specifications set out in the initial design phase.

Overall, the To-Do List application is a valuable tool that can help users manage their time and tasks more effectively. Whether used for personal or professional purposes, this application can make a significant difference in productivity and organization. With further development and improvements, this application has the potential to become even more useful and widely used.