**Enhanced To-Do List Application**

# Abstract

The Enhanced To-Do List Application is a desktop-based Java project designed to help users manage their daily tasks efficiently. The application allows users to add, delete, and mark tasks as completed while also setting reminders using an inbuilt alarm feature. Additionally, it incorporates attractive icons for various categories and a background image for better user experience.

# Introduction

Task management is an essential part of daily life for students and professionals alike. This To-Do List application provides a simple yet efficient way to organize and track tasks. Developed using Java Swing, it combines user-friendly design and useful functionalities such as setting alarms and categorizing tasks with icons.

# Existing System

Existing task management systems like basic notepads or simple list apps do not offer interactive features such as categorized tasks, icons, or reminders. They also lack an engaging graphical user interface. Users often forget tasks since there is no notification or alert mechanism.

# Proposed System

The proposed Enhanced To-Do List Application introduces alarm-based reminders that notify the user about upcoming tasks. Each task can be categorized under different types such as Shopping, Study, or Exercise, each represented by an icon. Users can easily add, delete, or mark tasks as completed, and the interface is enhanced with background images to make it visually appealing.

# Hardware Requirements

• Processor: Intel Core i3 or higher  
• RAM: Minimum 2 GB  
• Hard Disk: 200 MB of free space  
• Display: 1024x768 resolution or higher

# Software Requirements

• Operating System: Windows / Linux / macOS  
• Programming Language: Java (JDK 8 or above)  
• IDE: Eclipse / IntelliJ IDEA / NetBeans  
• Libraries: javax.swing, java.awt

# Source Code

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.\*;  
import java.time.\*;  
import java.util.\*;  
import javax.swing.Timer;  
  
public class ToDoListApp extends JFrame implements ActionListener {  
  
 // GUI Components  
 private DefaultListModel<Task> taskListModel;  
 private JList<Task> taskList;  
 private JTextField taskInput;  
 private JButton addButton, deleteButton, doneButton;  
 private JComboBox<String> typeComboBox;  
 private JSpinner hourSpinner, minuteSpinner;  
 private JLabel backgroundLabel;  
  
 // Icons for task types  
 private Map<String, ImageIcon> typeIcons;  
  
 public ToDoListApp() {  
 setTitle("Enhanced To-Do List App");  
 setSize(550, 550);  
 setDefaultCloseOperation(EXIT\_ON\_CLOSE);  
 setLayout(new BorderLayout());  
  
 // Load icons  
 typeIcons = new HashMap<>();  
 typeIcons.put("Shopping", new ImageIcon(new ImageIcon("shopping.png")  
 .getImage().getScaledInstance(25, 25, Image.SCALE\_SMOOTH)));  
 typeIcons.put("Study", new ImageIcon(new ImageIcon("study.png")  
 .getImage().getScaledInstance(25, 25, Image.SCALE\_SMOOTH)));  
 typeIcons.put("Exercise", new ImageIcon(new ImageIcon("exercise.png")  
 .getImage().getScaledInstance(25, 25, Image.SCALE\_SMOOTH)));  
  
 // Background label  
 backgroundLabel = new JLabel(new ImageIcon(new ImageIcon("background.jpg")  
 .getImage().getScaledInstance(550, 550, Image.SCALE\_SMOOTH)));  
 backgroundLabel.setLayout(new BorderLayout());  
 add(backgroundLabel);  
  
 // Input Panel  
 JPanel inputPanel = new JPanel(new FlowLayout());  
 inputPanel.setOpaque(false); // Transparent panel over background  
  
 taskInput = new JTextField(12);  
 typeComboBox = new JComboBox<>(new String[]{"Shopping", "Study", "Exercise"});  
 addButton = new JButton("Add Task");  
 deleteButton = new JButton("Delete Task");  
 doneButton = new JButton("Mark Done");  
  
 // Time spinners  
 hourSpinner = new JSpinner(new SpinnerNumberModel(0, 0, 23, 1));  
 minuteSpinner = new JSpinner(new SpinnerNumberModel(0, 0, 59, 1));  
  
 inputPanel.add(new JLabel("Task:"));  
 inputPanel.add(taskInput);  
 inputPanel.add(new JLabel("Type:"));  
 inputPanel.add(typeComboBox);  
 inputPanel.add(new JLabel("Hour:"));  
 inputPanel.add(hourSpinner);  
 inputPanel.add(new JLabel("Min:"));  
 inputPanel.add(minuteSpinner);  
 inputPanel.add(addButton);  
 inputPanel.add(deleteButton);  
 inputPanel.add(doneButton);  
  
 addButton.addActionListener(this);  
 deleteButton.addActionListener(this);  
 doneButton.addActionListener(this);  
  
 backgroundLabel.add(inputPanel, BorderLayout.NORTH);  
  
 // Task list  
 taskListModel = new DefaultListModel<>();  
 taskList = new JList<>(taskListModel);  
 taskList.setCellRenderer(new TaskCellRenderer());  
  
 JScrollPane scrollPane = new JScrollPane(taskList);  
 scrollPane.setOpaque(false);  
 scrollPane.getViewport().setOpaque(false);  
 backgroundLabel.add(scrollPane, BorderLayout.CENTER);  
  
 setLocationRelativeTo(null);  
  
 // Timer for alarms  
 Timer timer = new Timer(1000, e -> checkAlarms());  
 timer.start();  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 String command = e.getActionCommand();  
 String name = taskInput.getText().trim();  
 String type = (String) typeComboBox.getSelectedItem();  
 int hour = (Integer) hourSpinner.getValue();  
 int minute = (Integer) minuteSpinner.getValue();  
 LocalTime reminderTime = LocalTime.of(hour, minute);  
  
 switch (command) {  
 case "Add Task":  
 if (!name.isEmpty()) {  
 Task task = new Task(name, type, reminderTime, typeIcons.get(type));  
 taskListModel.addElement(task);  
 taskInput.setText("");  
 } else {  
 JOptionPane.showMessageDialog(this, "Please enter a task!");  
 }  
 break;  
  
 case "Delete Task":  
 int selected = taskList.getSelectedIndex();  
 if (selected != -1) {  
 taskListModel.remove(selected);  
 } else {  
 JOptionPane.showMessageDialog(this, "Select a task to delete!");  
 }  
 break;  
  
 case "Mark Done":  
 int idx = taskList.getSelectedIndex();  
 if (idx != -1) {  
 Task task = taskListModel.get(idx);  
 task.done = true;  
 taskList.repaint();  
 } else {  
 JOptionPane.showMessageDialog(this, "Select a task to mark done!");  
 }  
 break;  
 }  
 }  
  
 private void checkAlarms() {  
 LocalTime now = LocalTime.now();  
 for (int i = 0; i < taskListModel.size(); i++) {  
 Task task = taskListModel.get(i);  
 if (!task.done && task.reminderTime != null &&  
 task.reminderTime.getHour() == now.getHour() &&  
 task.reminderTime.getMinute() == now.getMinute()) {  
  
 JOptionPane.showMessageDialog(this,  
 "Reminder: " + task.name,  
 "Alarm",  
 JOptionPane.INFORMATION\_MESSAGE);  
 task.done = true;  
 taskList.repaint();  
 }  
 }  
 }  
  
 public static void main(String[] args) {  
 SwingUtilities.invokeLater(() -> new ToDoListApp().setVisible(true));  
 }  
  
 // Inner class: Task  
 static class Task {  
 String name, type;  
 boolean done = false;  
 LocalTime reminderTime;  
 ImageIcon icon;  
  
 Task(String name, String type, LocalTime reminderTime, ImageIcon icon) {  
 this.name = name;  
 this.type = type;  
 this.reminderTime = reminderTime;  
 this.icon = icon;  
 }  
  
 @Override  
 public String toString() {  
 return name + (done ? " ✓" : "");  
 }  
 }  
  
 // Custom Renderer  
 static class TaskCellRenderer extends JLabel implements ListCellRenderer<Task> {  
 @Override  
 public Component getListCellRendererComponent(JList<? extends Task> list, Task value, int index, boolean isSelected, boolean cellHasFocus) {  
 setText(value.toString());  
 setIcon(value.icon);  
 setOpaque(true);  
 setFont(new Font("Arial", Font.BOLD, 14));  
  
 if (value.done) setForeground(Color.GREEN.darker());  
 else setForeground(Color.BLACK);  
  
 if (isSelected) setBackground(Color.LIGHT\_GRAY);  
 else setBackground(new Color(255, 255, 255, 200));  
  
 return this;  
 }  
 }  
}

# Output

-------------------------------------------------------

| Task: [ Buy groceries ] Type: [ Shopping ▼ ] |

| Hour: [17] Min: [30] [Add Task] [Delete] [Done] |

-------------------------------------------------------

| 🛒 Buy groceries |

| 📚 Prepare for Java exam |

-------------------------------------------------------

(Popup example)

-----------------------------------

| Reminder: Buy groceries |

| [ OK ] |

-----------------------------------

# Conclusion

The Enhanced To-Do List Application successfully integrates task scheduling, reminders, and graphical enhancements in one platform. It demonstrates the use of Java Swing for GUI-based applications and can be further expanded with features such as cloud synchronization or mobile app integration.