**PERFECT CODE OF TO DO APPLICATION**

import tkinter as tk

import random

from tkinter.simpledialog import askinteger

class Task:

def \_\_init\_\_(self, description, due\_date, priority):

self.description = description

self.due\_date = due\_date

self.priority = priority

self.completed = False

class ToDoApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("To-Do List App")

self.root.configure(bg="lightblue") # Set the background color of the main window

self.task\_list = []

self.action\_log = []

# Create and layout GUI components

self.description\_label = tk.Label(root, text="Description:")

self.description\_entry = tk.Entry(root, width=50)

self.due\_date\_label = tk.Label(root, text="Due Date:")

self.due\_date\_entry = tk.Entry(root)

self.priority\_label = tk.Label(root, text="Priority:")

self.priority\_entry = tk.Entry(root)

self.add\_button = tk.Button(root, text="Add Task", command=self.add\_task)

self.delete\_button = tk.Button(root, text="Delete Task", command=self.delete\_task)

self.display\_button = tk.Button(root, text="Display Tasks", command=self.display\_tasks)

self.task\_listbox = tk.Listbox(root, width=70, height=10)

self.action\_logbox = tk.Listbox(root, width=70, height=10)

# Pack GUI components

self.description\_label.pack()

self.description\_entry.pack()

self.due\_date\_label.pack()

self.due\_date\_entry.pack()

self.priority\_label.pack()

self.priority\_entry.pack()

self.add\_button.pack()

self.delete\_button.pack()

self.display\_button.pack()

self.task\_listbox.pack()

self.action\_logbox.pack()

self.random\_colors = {} # Store random button colors

def add\_task(self):

description = self.description\_entry.get()

due\_date = self.due\_date\_entry.get()

priority = self.priority\_entry.get()

new\_task = Task(description, due\_date, priority)

self.task\_list.append(new\_task)

self.clear\_input\_fields()

# Set a random background color for the "Add Task" button

self.set\_random\_button\_color(self.add\_button)

# Log the action

self.log\_action(f"Task added: {description}")

def delete\_task(self):

if not self.task\_list:

print("No tasks to delete.")

return

task\_index = askinteger("Delete Task", "Enter the task number to delete:")

if task\_index is not None:

task\_index -= 1 # Adjust to 0-based index

if 0 <= task\_index < len(self.task\_list):

deleted\_task = self.task\_list.pop(task\_index)

# Set a random background color for the "Delete Task" button

self.set\_random\_button\_color(self.delete\_button)

# Log the action

self.log\_action(f"Task deleted: {deleted\_task.description}")

def log\_action(self, action):

self.action\_log.append(action)

self.action\_logbox.insert(tk.END, action)

def set\_random\_button\_color(self, button):

color = self.generate\_random\_color()

button.config(bg=color)

def generate\_random\_color(self):

# Generate a random hexadecimal color code

return "#{:02x}{:02x}{:02x}".format(

random.randint(0, 255),

random.randint(0, 255),

random.randint(0, 255)

)

def display\_tasks(self):

self.task\_listbox.delete(0, tk.END)

for idx, task in enumerate(self.task\_list, start=1):

task\_info = f"{idx}. Description: {task.description}, Due Date: {task.due\_date}, Priority: {task.priority}"

self.task\_listbox.insert(tk.END, task\_info)

def clear\_input\_fields(self):

self.description\_entry.delete(0, tk.END)

self.due\_date\_entry.delete(0, tk.END)

self.priority\_entry.delete(0, tk.END)

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = ToDoApp(root)

root.mainloop()

2ND PERFECT CODE

import tkinter as tk

from tkinter import simpledialog

class Task:

def \_\_init\_\_(self, description, due\_date, priority):

self.description = description

self.due\_date = due\_date

self.priority = priority

self.completed = False

class ToDoApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("To-Do List App")

self.task\_list = []

# Create and layout GUI components

self.description\_label = tk.Label(root, text="Description:")

self.description\_entry = tk.Entry(root, width=50) # Increase the width to display full task description

self.due\_date\_label = tk.Label(root, text="Due Date:")

self.due\_date\_entry = tk.Entry(root)

self.priority\_label = tk.Label(root, text="Priority:")

self.priority\_entry = tk.Entry(root)

self.add\_button = tk.Button(root, text="Add Task", command=self.add\_task)

self.delete\_button = tk.Button(root, text="Delete Task", command=self.delete\_task)

self.task\_listbox = tk.Listbox(root, width=70, height=10) # Increase the width and height

self.display\_button = tk.Button(root, text="Display Tasks", command=self.display\_tasks)

# Pack GUI components

self.description\_label.pack()

self.description\_entry.pack()

self.due\_date\_label.pack()

self.due\_date\_entry.pack()

self.priority\_label.pack()

self.priority\_entry.pack()

self.add\_button.pack()

self.delete\_button.pack()

self.task\_listbox.pack()

self.display\_button.pack()

def add\_task(self):

description = self.description\_entry.get()

due\_date = self.due\_date\_entry.get()

priority = self.priority\_entry.get()

new\_task = Task(description, due\_date, priority)

self.task\_list.append(new\_task)

print("Task added successfully.")

self.clear\_input\_fields()

self.display\_tasks()

def delete\_task(self):

if not self.task\_list:

print("No tasks to delete.")

return

task\_index = simpledialog.askinteger("Delete Task", "Enter the task number to delete:")

if task\_index is not None:

task\_index -= 1 # Adjust to 0-based index

if 0 <= task\_index < len(self.task\_list):

self.task\_list.pop(task\_index)

self.display\_tasks()

print("Task deleted successfully.")

else:

print("Invalid task number.")

def display\_tasks(self):

self.task\_listbox.delete(0, tk.END)

for idx, task in enumerate(self.task\_list, start=1):

task\_info = f"{idx}. Description: {task.description}, Due Date: {task.due\_date}, Priority: {task.priority}"

self.task\_listbox.insert(tk.END, task\_info)

def clear\_input\_fields(self):

self.description\_entry.delete(0, tk.END)

self.due\_date\_entry.delete(0, tk.END)

self.priority\_entry.delete(0, tk.END)

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = ToDoApp(root)

root.mainloop()