



Kabul\_University\_Afghanistan  
Team Project for our final exam  
Project name (TO-DO-LIST)  
Submission day: 2025/11/19

Team members: Ali Sher Rezaie, Enayatullah Habibi  
Mohammad Hassani, Mohebullah Sarwari

# To Do List

First of all I am going to say who are we that made this To Do List.

We are students of Kabul University at ISE(Information and Science Engineering) field, 2025 is our first year at university. Our teacher has given us job to make a To Do List for our final project. Now it is time to say the most important points of our project.

## Key Features:

1. Modern user interface with dark theme
2. Persistent storage using JSON files
3. Full task management (add, delete, mark as done)
4. Visual representation of task status
5. Security confirmations for sensitive actions

## Technical Components:

- customtkinter – for advanced GUI
- JSON – for data storage
- Event Handling – to manage user interactions
- Lambda Functions – for dynamic bindings
- List Comprehensions – for data filtering

This program offers a complete and user-friendly solution for managing daily tasks.

Here is our code:

```
#  
=====  
=  
  
# TO-DO LIST PRO - GUI VERSION 📱  
# Author: Levi / Streamlined by AI Assistant  
#  
=====  
=  
  
import customtkinter as ctk  
from tkinter import messagebox  
import json, os  
  
DATA_FILE = "tasks.json"  
  
# --- Helper Functions ---  
def generate_new_id(tasks):  
    """Generates a unique ID."""  
    if not tasks: return 1  
    max_id = max(task.get('id', 0) for task in tasks)  
    return max_id + 1  
  
def load_tasks():  
    """Loads tasks safely."""  
    if os.path.exists(DATA_FILE):  
        try:            with open(DATA_FILE, "r") as file:  
                tasks = json.load(file)  
        except json.JSONDecodeError:  
            tasks = []  
    else:  
        tasks = []  
    return tasks
```

```
with open(DATA_FILE, "r") as f:
    data = json.load(f)
    if isinstance(data, list): return data
except (json.JSONDecodeError, IOError):
    print("Error loading tasks. Starting fresh.")
return []

def save_tasks(tasks):
    """Saves tasks safely."""
    try:
        with open(DATA_FILE, "w") as f:
            json.dump(tasks, f, indent=4)
    except IOError as e:
        messagebox.showerror("Save Error", f"Could not save tasks: {e}")

# --- App Class ---
class ToDoApp(ctk.CTk):
    def __init__(self):
        super().__init__()
        self.title("◆ To-Do List Pro ◆")
        self.geometry("500x600")
        ctk.set_appearance_mode("dark")
        ctk.set_default_color_theme("blue")
        self.tasks = load_tasks()
```

```
# UI Layout

    ctk.CTkLabel(self, text="📝 To-Do List Pro", font=("Segoe UI", 28,
"bold")).pack(pady=20)

    self.entry_frame = ctk.CTkFrame(self, fg_color="transparent")
    self.entry_frame.pack(pady=10, padx=10, fill="x")

        self.task_entry = ctk.CTkEntry(self.entry_frame, placeholder_text="Enter
new task...", height=40)
        self.task_entry.pack(side="left", fill="x", expand=True, padx=(0, 10))
        self.task_entry.bind('<Return>', lambda e: self.add_task())
        ctk.CTkButton(self.entry_frame, text="Add Task", width=100,
command=self.add_task).pack(side="right")

    self.task_frame = ctk.CTkScrollableFrame(self, width=450, height=400,
fg_color="#1a1a1a")
    self.task_frame.pack(pady=15, padx=10, fill="both", expand=True)

    self.bottom_frame = ctk.CTkFrame(self, fg_color="transparent")
    self.bottom_frame.pack(pady=10)

        ctk.CTkButton(self.bottom_frame, text="🧹 Clear Done Tasks", width=120,
fg_color="orange", command=self.clear_done_tasks).grid(row=0, column=0,
padx=10)
        ctk.CTkButton(self.bottom_frame, text="💾 Save & Exit", width=120,
fg_color="green", command=self.save_and_exit).grid(row=0, column=1,
padx=10)
```

```
    self.display_tasks()

# --- Methods ---

def display_tasks(self):
    """Renders tasks, clearing previous display."""
    for widget in self.task_frame.winfo_children(): widget.destroy()

    for task in self.tasks:
        row = ctk.CTkFrame(self.task_frame, fg_color="#333333",
                           corner_radius=8)
        row.pack(fill="x", pady=4, padx=5)

        font_style = ("Segoe UI", 14, "overstrike") if task["done"] else ("Segoe
UI", 14)
        text_color = "gray" if task["done"] else "white"

        check_var = ctk.BooleanVar(value=task["done"])
        checkbox = ctk.CTkCheckBox(row, text=task["title"], variable=check_var,
                                   font=font_style, text_color=text_color,
                                   command=lambda tid=task['id']: self.toggle_done(tid))
        checkbox.pack(side="left", padx=10, pady=10)

        ctk.CTkButton(row, text="X", width=30, fg_color="red",
                      command=lambda tid=task['id']: self.delete_task(tid)).pack(side="right", padx=10)

def add_task(self):
```

```
title = self.task_entry.get().strip()
if not title:
    messagebox.showwarning("Warning", "Please enter a task.")
    return

new_task = {"id": generate_new_id(self.tasks), "title": title, "done": False}
self.tasks.append(new_task)
save_tasks(self.tasks)
self.task_entry.delete(0, "end")
self.display_tasks()

def delete_task(self, task_id):
    self.tasks = [task for task in self.tasks if task['id'] != task_id]
    save_tasks(self.tasks)
    self.display_tasks()

def toggle_done(self, task_id):
    # Find task by ID and flip status
    for task in self.tasks:
        if task['id'] == task_id:
            task["done"] = not task["done"]
            break
    save_tasks(self.tasks)
    self.display_tasks() # Redraw to update visual styling
```

```

def clear_done_tasks(self):
    if messagebox.askyesno("Confirm Clear Done", "Clear all completed tasks?"):
        self.tasks = [task for task in self.tasks if not task["done"]]
        save_tasks(self.tasks)
        self.display_tasks()

```

```

def save_and_exit(self):
    save_tasks(self.tasks)
    self.destroy()

```

# --- Run ---

```

if __name__ == "__main__":
    app = ToDoApp()
    app.mainloop()

```

```

# To Do List.py
=====
# TO-DO LIST PRO - GUI VERSION
# Author: Levi / Streamlined by AI Assistant
# =====

import customtkinter as ctk
from tkinter import messagebox
import json, os

DATA_FILE = "tasks.json"

# --- Helper Functions ---
def generate_new_id(tasks):
    """Generates a unique ID."""
    if not tasks: return 1
    max_id = max(task.get('id', 0) for task in tasks)
    return max_id + 1

def load_tasks():
    """Loads tasks safely."""
    if os.path.exists(DATA_FILE):
        try:
            with open(DATA_FILE, "r") as f:
                data = json.load(f)
                if isinstance(data, list): return data
        except (json.JSONDecodeError, IOError):
            print("Error loading tasks. Starting fresh.")
    return []

def save_tasks(tasks):
    """Saves tasks safely."""

```

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} Python

```
1  #!/usr/bin/python3
2
3  DATA_FILE = "tasks.json"
4
5  import json
6  import tkinter as tk
7  from tkinter import messagebox
8  from tkinter import ttk
9  from tkinter import filedialog
10
11  def load_tasks():
12      try:
13          with open(DATA_FILE, "r") as f:
14              tasks = json.load(f)
15      except FileNotFoundError:
16          tasks = []
17
18  def save_tasks(tasks):
19      """Saves tasks safely."""
20
21      try:
22          with open(DATA_FILE, "w") as f:
23              json.dump(tasks, f, indent=4)
24      except IOError as e:
25          messagebox.showerror("Save Error", f"Could not save tasks: {e}")
26
27
28  # --- App Class ---
29  class ToDoApp(ctk.CTk):
30
31      def __init__(self):
32          super().__init__()
33          self.title("❖ To-Do List Pro ❖")
34          self.geometry("500x600")
35          ctk.set_appearance_mode("dark")
36          ctk.set_default_color_theme("blue")
37          self.tasks = load_tasks()
38
39
40      # UI Layout
41      ctk.CTkLabel(self, text="❖ To-Do List Pro", font=("Segoe UI", 28, "bold")).pack(pady=20)
42
43      self.entry_frame = ctk.CTkFrame(self, fg_color="transparent")
44      self.entry_frame.pack(pady=10, padx=10, fill="x")
45
46
47      self.task_entry = ctk.CTkEntry(self.entry_frame, placeholder_text="Enter new task...", height=40)
48      self.task_entry.pack(side="left", fill="x", expand=True, padx=(0, 10))
49      self.task_entry.bind('<Return>', lambda e: self.add_task())
50
51      ctk.CTkButton(self.entry_frame, text="Add Task", width=100, command=self.add_task).pack(side="right")
52
53
54      self.task_frame = ctk.CTkScrolledFrame(self, width=450, height=400, fg_color="#1a1a1a")
55
56
57
58
59
```

Ln 58, Col 1 Spaces: 4 UTF-8 CRLF {} Python ⚙️ 🔍

```
1  self.task_frame = ctk.CTkScrolledFrame(self, width=450, height=400, fg_color="#1a1a1a")
2  self.task_frame.pack(pady=15, padx=10, fill="both", expand=True)
3
4
5  self.bottom_frame = ctk.CTkFrame(self, fg_color="transparent")
6  self.bottom_frame.pack(pady=10)
7
8
9  ctk.CTkButton(self.bottom_frame, text="✗ Clear Done Tasks", width=120, fg_color="orange", command=self.clear_done_tasks).grid(row=0, column=0)
10 ctk.CTkButton(self.bottom_frame, text="💾 Save & Exit", width=120, fg_color="green", command=self.save_and_exit).grid(row=0, column=1)
11
12
13  self.display_tasks()
14
15
16  # --- Methods ---
17  def display_tasks(self):
18      """Renders tasks, clearing previous display."""
19
20      for widget in self.task_frame.winfo_children():
21          widget.destroy()
22
23
24      for task in self.tasks:
25
26          row = ctk.CTkFrame(self.task_frame, fg_color="#333333", corner_radius=8)
27          row.pack(fill="x", pady=4, padx=5)
28
29
30          font_style = ("Segoe UI", 14, "overstrike") if task["done"] else ("Segoe UI", 14)
31          text_color = "gray" if task["done"] else "white"
32
33
34          check_var = ctk.BooleanVar(value=task["done"])
35          checkbox = ctk.CTkCheckBox(row, text=task["title"], variable=check_var,
36                                     font=font_style, text_color=text_color,
37                                     command=lambda tid=task['id']: self.toggle_done(tid))
38
39          checkbox.pack(side="left", padx=10, pady=10)
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
```

Ln 86, Col 28 Spaces: 4 UTF-8 CRLF {} Python ⚙️ 🔍

The screenshot shows a Python code editor interface with a dark theme. The main window displays a file named 'To Do List.py' containing Python code for a GUI application. The code includes functions for adding tasks, deleting tasks, toggling task completion status, and clearing completed tasks. It also handles saving tasks to a file and running the application's main loop. The code editor has various toolbars and status bars at the top and bottom.

```
85         command=lambda tid=task['id']: self.toggle_done(tid))
86         checkbox.pack(side="left", padx=10, pady=10)
87
88         ctk.CTkButton(row, text="X", width=30, fg_color="red", command=lambda tid=task['id']: self.delete_task(tid)).pack(side="right")
89
90     def add_task(self):
91         title = self.task_entry.get().strip()
92         if not title:
93             messagebox.showwarning("Warning", "Please enter a task.")
94             return
95
96         new_task = {"id": generate_new_id(self.tasks), "title": title, "done": False}
97         self.tasks.append(new_task)
98         save_tasks(self.tasks)
99         self.task_entry.delete(0, "end")
100        self.display_tasks()
101
102    def delete_task(self, task_id):
103        self.tasks = [task for task in self.tasks if task['id'] != task_id]
104        save_tasks(self.tasks)
105        self.display_tasks()
106
107    def toggle_done(self, task_id):
108        # Find task by ID and flip status
109        for task in self.tasks:
110            if task['id'] == task_id:
111                task["done"] = not task["done"]
112                break
113        save_tasks(self.tasks)
114        self.display_tasks() # Redraw to update visual styling
115
116    def clear_done_tasks(self):
117        if messagebox.askyesno("Confirm Clear Done", "Clear all completed tasks?"):
118            self.tasks = [task for task in self.tasks if not task["done"]]
119            save_tasks(self.tasks)
120            self.display_tasks()
121
122    def save_and_exit(self):
123        save_tasks(self.tasks)
124        self.destroy()
125
126    # --- Run ---
127 if __name__ == "__main__":
128     app = ToDoApp()
129     app.mainloop()
130
131
```

Ln 85, Col 28 Spaces: 4 UTF-8 CRLF () Python ⚙️

This screenshot shows the same Python code editor interface as the first one, but with more methods added to the class. The new methods include 'clear\_done\_tasks' which clears all completed tasks, 'save\_and\_exit' which saves tasks and exits the application, and a run block at the bottom. The code remains largely the same as the first screenshot, with the addition of these new methods.

```
102    def delete_task(self, task_id):
103        self.tasks = [task for task in self.tasks if task['id'] != task_id]
104        save_tasks(self.tasks)
105        self.display_tasks()
106
107    def toggle_done(self, task_id):
108        # Find task by ID and flip status
109        for task in self.tasks:
110            if task['id'] == task_id:
111                task["done"] = not task["done"]
112                break
113        save_tasks(self.tasks)
114        self.display_tasks() # Redraw to update visual styling
115
116    def clear_done_tasks(self):
117        if messagebox.askyesno("Confirm Clear Done", "Clear all completed tasks?"):
118            self.tasks = [task for task in self.tasks if not task["done"]]
119            save_tasks(self.tasks)
120            self.display_tasks()
121
122    def save_and_exit(self):
123        save_tasks(self.tasks)
124        self.destroy()
125
126    # --- Run ---
127 if __name__ == "__main__":
128     app = ToDoApp()
129     app.mainloop()
130
131
```

Ln 131, Col 3 Spaces: 4 UTF-8 CRLF () Python ⚙️

Here the main and graphical page of To Do List.

