Rizvi College of Engineering

Department of Computer Engineering

SKL-PP Mini Project Report

On

TO-DO LIST

Submitted

by

Utsav Kuntalwad (22)

Mayur Kyatham (23)

Srushti Sawant (37)

University of Mumbai (2021 –2022)

ABSTRACT

TODO List are the lists that we generally use to maintain our day to day tasks or list of everything that we have to do, with the most important tasks at the top of the list, and the least important tasks at the bottom. It is helpful in planning our daily schedules. We can add more tasks any time and delete a task which is completed. The four major tasks that we can perform in a TODO list are:

1. Add tasks
2. Update tasks
3. Read tasks
4. Delete tasks

We are going to implement a simple python to-do list in which we can add a task and delete it when it's done. We will be using a Python package called Tkinter which is a widely used Python GUI library. It is shipped with python, so we do not have to download or install it separately, we can just import and start with it.

A GUI is basically a medium to interact and present information to the users. One of the major advantages of using Tkinter is that it works well on any machine be it windows, linux, or macOS.

Certificate

This is to certify that the mini project report entitled "**TO-DO LIST**" has been submitted by **Utsav Kuntalwad , Mayur Kyatham , Srushti Sawant** under the guidance of Prof. **Shaikh Mohd Ashfaque** in partial fulfillment of the requirement for the award of 2nd year Engineering in Computer Engineering from University of Mumbai.

Certified By

Prof. Shaikh Mohd Ashfaque	Prof. Shiburaj P.
Project Guide	Head of Department
Prof. _____	Prof. _____
Internal Examiner	External Examiner

Dr. Varsha Shah

Principal

Department of Computer Engineering

Rizvi College of Engineering,

Off Carter Road, Bandra (W), Mumbai-400050

Index

Topic	Page no.
Abstract	02
Table of contents	04
Project Description	05
Methodology/ Block Diagram	07
Software & Hardware Requirements	11
Project Code	12
Result / Output	15
Conclusion	17
References	18

PROJECT DESCRIPTION

What is a To-Do List?

What is a To-Do List? The definition is a simple one. ***It's a list of tasks you need to complete or things that you want to do.***

Most typically, they're organised in order of priority. Traditionally, they're written on a piece of paper or post it notes and act as a memory aid. As technology has evolved we have been able to create a to-do lists with excel spreadsheets, word documents, e mail lists, to-do list apps, Microsoft to do and google to do list to name a few. You can use a to do list in your home and personal life, or in the workplace.

Having a list of everything you need to do written down in one place means you shouldn't forget anything important. By prioritizing the tasks in the list you plan the order in which you're going to do them and can quickly see what needs your immediate attention and what tasks you can leave until a little later.

The Benefits of Using a To-Do List

One of the most important reasons you should use a to do list is that it will help you stay organized. When you write all your tasks in a list, they seem more manageable. When you've got a clear outline of the tasks you've got to do and those you've completed, it helps you stay focused. While freeing up space in your mind for other more creative tasks.

When you complete a task, you can cross it off your list. This gives you a sense of progress and achievement, something you'll lack if you're always rushing from one task to the next. If you feel a sense of achievement, it spurs you on and motivates you to keep moving forward.

But that's not the only benefit of a to do list. Here are a few more:

Improves your memory: A to do list acts as an external memory aid. It's only possible to hold a few pieces of information at one time. Keep a to do list and you'll be able to keep track of everything, rather than just a few of the tasks you need to do. Your to do list will also reinforce the information, which makes it less likely you're going to forget something.

Increases productivity: A to do list allows you to prioritize the tasks that are more important. This means you don't waste time on tasks that don't require your

immediate attention. Your list will help you stay focused on the tasks that are the most important.

Helps with motivation: To do lists are a great motivational tool because you can use them to clarify your goals. You can divide your long-term goal into smaller, more achievable short-term goals and as you tick each one off your list, your confidence will increase.

What is a To-Do List in business and why is it important?

It seems such a simple solution by putting pen to paper and taking time out of your day to create a to do list, a plan for your day helps define your challenges and goals. Preventing time from being wasted trying to identify what is the next most important task to tackle next and even more important makes sure you don't forget to do something important.

METHODOLOGY

Import Modules
Before we start using Tkinter, we need to call Tkinter for use. so we import the module. Here * means everything. So we are importing everything from Tkinter then in the second line we have imported message box from Tkinter

Create & Configure Window
After importing module, we will create a window so that we can place widgets on it.

- **ws** is used to initialize **Tk()**. From now **ws** will be called as the parent window. All other widgets will be placed on it.
- **ws.geometry('width x height + x-position+ y-position')**
  All the values provided must be integers.
  - the width refers to the horizontal space of the window.
  - height refers to the vertical space of the window.
  - x-position refers to the position of the window on the display over the x-axis.
  - y-position refers to the position of the window on the display over the y-axis.
- the **title** will add a title to the window. In our case, we have provided our website name as a title. You can find the title on the top left of the window next to feather.
- **config** is used to provide background color to the window.
- **resizable** accepts boolean values. Since the boolean values are provided false for both height and width that means the window can't be resized. To know more about resizable please read our blog on python Tkinter windows size.
- **ws.mainloop()** holds the screen so that we can see the window. It is an infinite loop. screen pops up and then disappears but with this infinite loop this process of appearing & disappearing keeps on happening very fast. And we keep on seeing the updated window.

Creating a frame

- **Frame** widgets are used to hold other widgets.
- They help in keeping & maintaining user interface (UI) & user experience (UX) clean & organized.
- Moving forward we will place Listbox, scrollbars & buttons inside the frame.
- So in this way frame will act as an additional window over the parent window.
- Another benefit of placing a frame is now we will add scrollbars to the frame and that solves our purpose.
- Scrollbars are no easy to place but using frames we can do it in no time.
- **paddy=10** means we have added extra padding around the frame from outside.

Adding List box

In this section, we will learn why and how we have used List box on the window.

- **lb** is the variable name for storing List box.
- List box is placed on the frame window.
- **width:** Horizontal space provided is 25.
- **height:** 8 rows in the vertical position are provided.
- **font:** Times New Roman font is provided with 14 sizes.
- **bd = 0** refers to the border is zero
- **fg** is the foreground color or the text color.
- **highlightthickness=0** every time the focus is moved to any item then it should not show any movement that is value 0 value is provided. by default it has some value.
- **selectbackground** it decides the color of the focused item in the Listbox.
- **activestyle="none"** removes the underline that appears when the item is selected or focused.
- **geometry manager** used is **pack()**
- **side=LEFT**this will keep the List box to the left side of the frame. We did this on purpose so that we can assign the right position to scrollbars.
- **fill=BOTH** this will fill the blank space in both the directions that are x and y

Adding dummy data

- We have added dummy data so that the application is ready to view. You add or delete whatever data you want.
- the data is in a list format and is stored in a variable named **task_list**.
- **for loop** is used to insert data in the List box.
- every time loop runs it adds an item to the Listbox & this process keeps on going until all the items in the **task_list** are inserted.
- **lb.insert(END, item)** this command stacks the items in List box.
  - **lb** is the variable used for **List box**
  - **insert** is a built-in method of List box to insert data.
  - **END** signifies that a new item will be added in the end. If **END** is replaced with **0** then new data will be added at the top.
  - **item** is the list item from **task_list**

Adding Scrollbars

In this section, we will understand why and how scrollbars are added to the window.

- **Scrollbars** are used so that users can scroll the information that is placed in a limited size on the window.
- In this scrollbars are placed on a frame and the variable assigned is sb
- The **geometry** method used is a pack() so that everything remains dynamic and in a sequence.
- **side=RIGHT** we have placed scrollbars on the right side of the frame.
- In the above code, we have provided **side=LEFT** to List box. So in this way, both the widgets are assigned paralleled.
- **fill=BOTH** this will fill the blank space in both the directions that are x and y
- **lb.config(yscrollcommand=sb.set)** here we have assigned a purpose to the scrollbar. In other words, we have bind Listbox with scrollbar
- **sb.config(command=lb.yview)**, here yview means scrollbar will go in the vertical direction. If it would have been xview then the scrollbar would have worked in the horizontal direction.

Adding Entry Box

- **Entry box** is used to take input from the user.
- **ws**: entry box is placed on the parent window
- **font:** provides font name i.e ‘Times New Roman’ and size is 14
- **Geometry manager** used is **pack** with padding of 20 outside the widget

Adding another frame for buttons

Frames are used to organise the widgets. We have used separate frame for buttons.

Adding Buttons

- Buttons are placed to trigger some action when pressed.
- Here we have created two button (**addTask** & **deleteTask**). Both of them have same features and look accept the **background color** and **command.**

- **Command**: When button is clicked the the function mentioned in command is called. In this case if user clicks on **addTask_btn** button then **newTask function** is called & when user clicks on the **delTask_btn** Button then **delTask function** is called.

newTask() function

- In this function, we have stored the value of the entry box in the task variable
- get() method is used to pull the value provided by the user in the entry box.
- If-else condition is applied to avoid black space entry in the Listbox.
- if the task does not have blank space then only it will allow it to store the information in the Listbox otherwise it will display a warning message box informing the user that the entry box can't be empty.

deleteTask() function

- Here **ANCHOR** refers to the selected item in the Listbox.
- **lb** variable assigned to Listbox
- delete is a built-in function to delete Listbox item.
- User will select the item in the Listbox and then to trigger this function he/she will click on the **deleteTask button**. Immediately the item will disappear from the Listbox.

SOFTWARE & HARDWARE REQUIREMENTS

Required Software:

Necessary Software's/Libraries are:

I. Visual Studio Code

II. tkinter

III. pickel

Required Hardware:

- Processor (CPU) with 2 gigahertz (GHz) frequency or above
- A minimum of 4 GB of RAM
- A minimum of 50 GB of available space on the hard disk
- Internet Connection Broadband (high-speed) Internet connection with a speed of 4 Mbps or higher
- Windows system with 16 GB RAM
- CUDA dependencies for GPU acceleration
- CUDA-enabled Nvidia graphical processing unit (GPU)

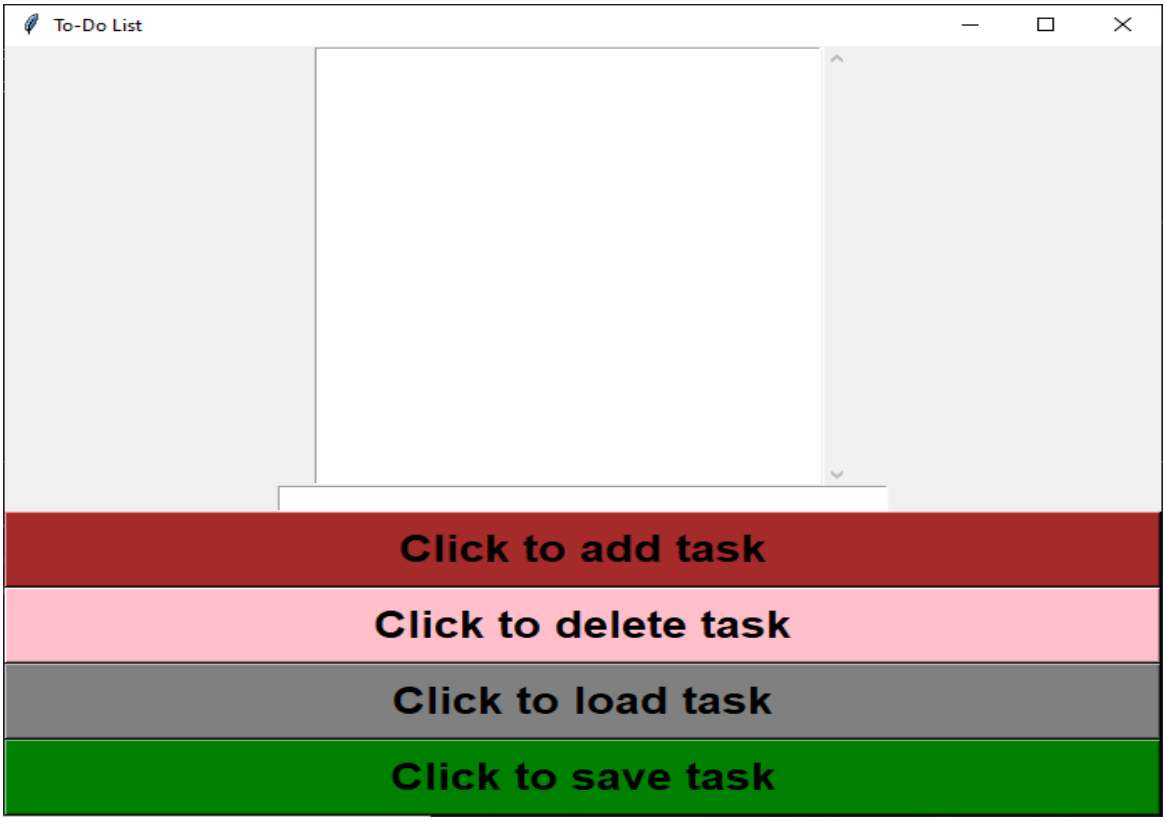
PROJECT CODE

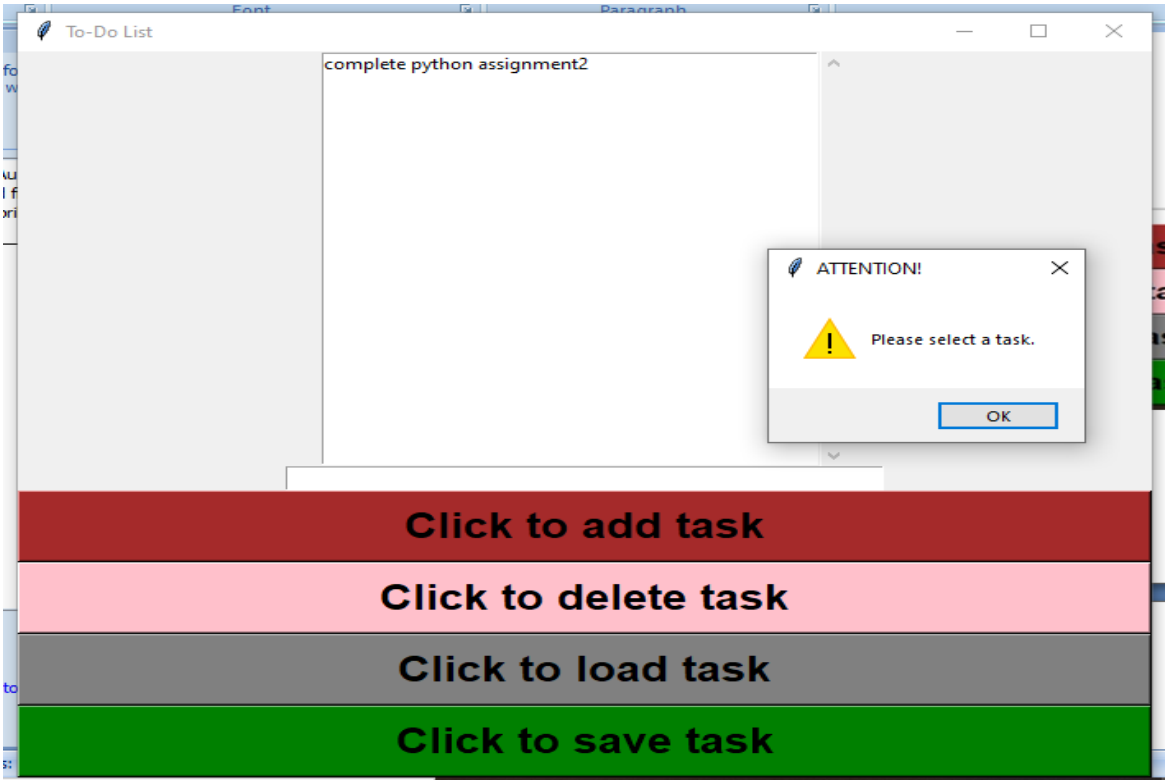
```
import tkinter
import tkinter.messagebox
import pickle
root = tkinter.Tk()
root.title("To-Do List")
def add_task():
    task = entry_task.get()
    if task != "":
        listbox_tasks.insert(tkinter.END, task)
        entry_task.delete(0, tkinter.END)
    else:
        tkinter.messagebox.showwarning(title="ATTENTION!",
message="Please enter a task.")
def delete_task():
    try:
        task_index = listbox_tasks.curselection()[0]
        listbox_tasks.delete(task_index)
    except:
        tkinter.messagebox.showwarning(title="ATTENTION!",
message="Please select a task.")
def load_tasks():
    try:
```

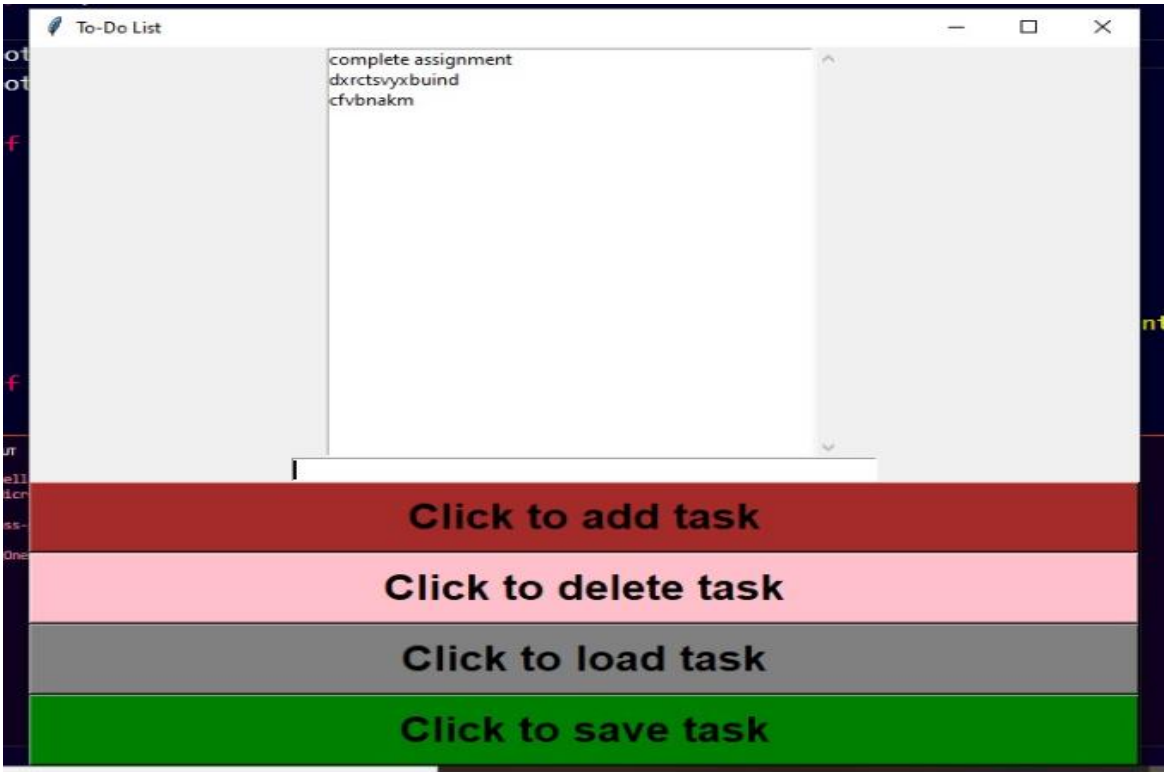
```
        tasks = pickle.load(open("tasks.dat", "rb"))
        listbox_tasks.delete(0, tkinter.END)
        for task in tasks:
            listbox_tasks.insert(tkinter.END, task)
    except:
        tkinter.messagebox.showwarning(title="ATTENTION!",
message="Cannot find tasks.dat.")
def save_tasks():
    tasks = listbox_tasks.get(0, listbox_tasks.size())
    pickle.dump(tasks, open("tasks.dat", "wb"))
frame_tasks = tkinter.Frame(root)
frame_tasks.pack()
listbox_tasks = tkinter.Listbox(frame_tasks, height=20, width=50)
listbox_tasks.pack(side=tkinter.LEFT)
scrollbar_tasks = tkinter.Scrollbar(frame_tasks)
scrollbar_tasks.pack(side=tkinter.RIGHT, fill=tkinter.Y)
listbox_tasks.config(yscrollcommand=scrollbar_tasks.set)
scrollbar_tasks.config(command=listbox_tasks.yview)
entry_task = tkinter.Entry(root, width=60)
entry_task.pack()
button_add_task = tkinter.Button(root, text="Click to add task",
font=("arial", 20,"bold"), background="brown", width=40,
command=add_task)
button_add_task.pack()
```

```
button_delete_task = tkinter.Button(root, text="Click to delete task", font=("arial", 20,"bold"), background="pink", width=40, command=delete_task)

button_delete_task.pack()

button_load_tasks = tkinter.Button(root, text="Click to load task", font=("arial", 20,"bold"), background="grey", width=40, command=load_tasks)

button_load_tasks.pack()

button_save_tasks = tkinter.Button(root, text="Click to save task", font=("arial", 20,"bold"), background="green", width=40, command=save_tasks)

button_save_tasks.pack()

root.mainloop()
```

RESULT / OUTPUT






To-Do List
complete assignment
dxrctsvyxbuind
cfvbnakm
Click to add task
Click to delete task
Click to load task
Click to save task

CONCLUSION

To-do list is a really neat digital to-do-list. It does everything you'd want a to-do-list to do: you can assign task priorities, group tasks by projects, set deadlines, and so on. Although I don't use To-do list as my primary organizing tool anymore, it's still my favorite standalone to-do list app.

First, you'll want to make a To-do list account. This is pretty simple, and won't take you long at all. Here, you'll see all the instructions you need to perform actions to your account using the API. You can install the API via pip using the shell command `$ pip install to-do list-python`.

Next, you'll need to find your To-do list account's API token. This is a unique string of characters that the API uses to identify your account. You can find this by logging into the To-do list web app, clicking on your profile picture in the top-right hand corner, and then selecting 'Integrations'; your API token is clearly marked on this screen.

Finally, I'd recommend playing around with To-do list a little bit if you're new to it. Seeing how the core features work might help to give you some ideas about what kind of tasks to automate later. At the very least, create a new project to store your automated tasks, and make a note of its ID.

REFERENCES

1. Dhanya Bibin,Madhu D.Nair,P.Punitha “Studying the To-Do list working Using Deep Belief Networks”, IEEE Access 2017

2. Surasak Kasetsirikul, Jirayut Buranapong, “The development of various plans schedule techniques: a review of the approaches with focus various methods”, Journal 2016 15:358

3. Mahdieh Poostchi, Kamolrat Silamut, Richard J Maude, Stefan Jaeger, George Thoma, “Image Analysis of To-Do list”, Translational Research Vol. 194, April 2014

4. Faza Maulaf Azif, Hanung Adi Nugroho, Sunu Wibirama, “Working of To-Do list”, Communications in Technology 3(1) (2018) 27- 35

5. Sairaj Bharadwaj Reddy & D. Sujitha Juliet, “Transfer Learning with ResNet-50 for To-Do list ”, International Conference on Communication and Signal Processing

6. Sivaramakrishnan Rajaraman, Kamolrat Silamut, Md. A. Hossain, I. Ersoy, Richard J. Maude Stefan Jaeger, George R. Thoma, Sameer K. Antani, “Understanding the logic of To-Do list images”, Journal of Imaging 5(3) 034501 August 2018