Republic of the Philippines
CAVITE STATE UNIVERSITY
Don Severino delas Alas Campus
Indang, Cavite

OBJECT-ORIENTED PROGRAMMING

Final Project
<TASK MONITORING>

Submitted by:
Tayab, Dirk M. – Leader

Aguado, Danielle Ysabelle M.
Albero, Justine Kate P.
Alegre, Ericka Jane A.
Bangay, Rhonan Richmond D.
Cubol, Elise Brixe S.
Rupido, Baby Angel E.
Serenio, Marc Jay D.
– Members

Date Submitted
17-06-2022

Submitted to:

Engr. Maria Rizette H. Sayo

I. Objectives

This Final project aims to:

- Apply the knowledge and skills learned in the previous topics in Object Oriented Programming;
- Demonstrate the different coding approaches covered in this course;
- Create a Task Monitoring GUI using Tkinter module;
- Aids everyone in becoming more organized in their work;
- Assist the user in practicing time management and maximizing time spent on their task.
- Show a task monitoring GUI with the appropriate output and functionality.
- To develop an understanding in establishing a task monitoring

II. Methods

This section provides the methods and descriptions used to develop the task monitoring GUI and the purpose of this project is to perform well on the accurate codes for this work.

a. Conceptualization / (Planning and Visualization)

The authors conducted an online meeting and discussed the design plan using Adobe Photoshop for better visualization of the target outcome of this project. The aspect ratio used in the task is only 700 px by 500 px for better output. There will be three input boxes, namely: Category, Task name and due date, with the outcome appearing below.

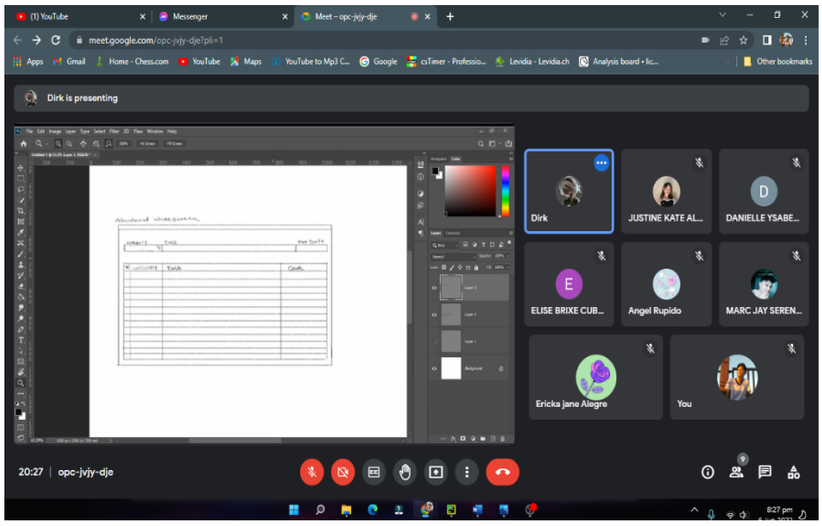

Figure 1. Planning and Visualization of the GUI

Figure 2. Desired layout of the GUI

Figure 3. GUI Name, Tagline, and Logo

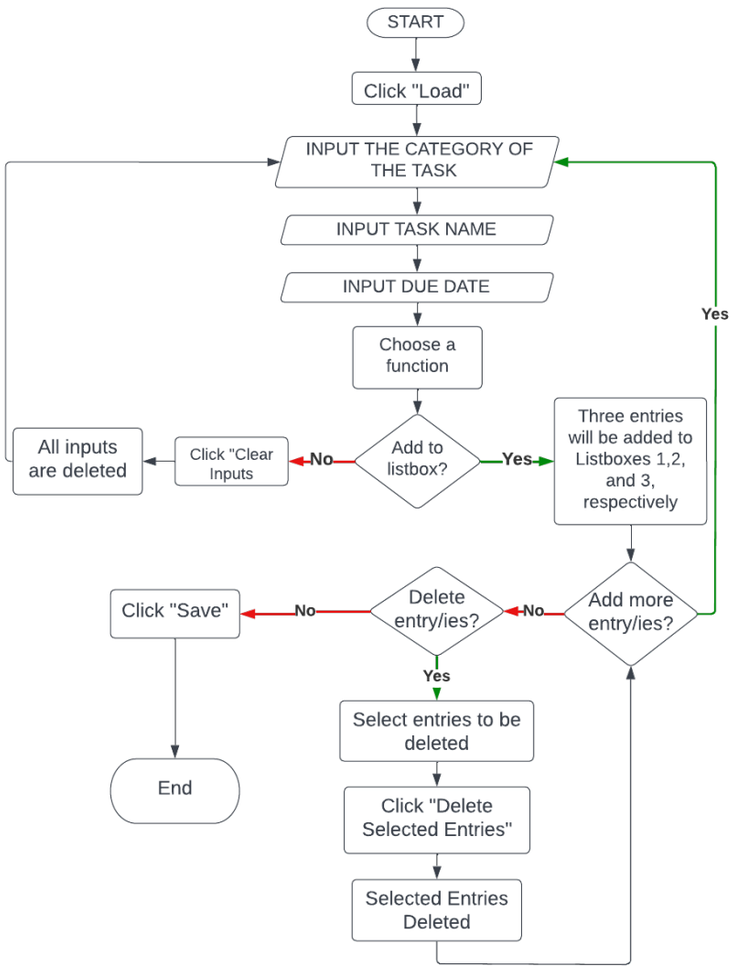

Figure 4. Flowchart of the GUI

b. Widgets

To be able to display and input information to the program widgets are added after the creation of the program window.

Entry Widget

It is a precondition to provide entry widgets for the users to be able to input data onto the application. The three widgets added to the program. Specifically, a ComboBox widget for the task category filled with BSCPE 1-1's current subjects as options and the ability for the user to create their own categories; A simple entry widget made to provide an input for the task's name or description of the tasks; Lastly, a DateEntry widget was added rather than an Entry widget for the date input as DateEntry provides a calendar to easily pick a given date.

Labels

As a way to recognize the function of the entry widgets in the program, each of them is labeled as "CATEGORY" for the ComboBox, "TASK NAME" for the Entry widget, and "DUE DATE" for the DateEntry widgets.

ListBoxes

To provide a space to list down the tasks and their information, a ListBox widget is added for the three entry widget. For some cases where there are a lot of entries listed on the ListBox, a scrollbar is added to the right edge of the widget.

Buttons

Several button widgets were added for the program to function properly. Firstly, an "Add" button was made to include the entries' contents to the ListBox. For removing a task from the ListBox a "Delete" button was made. And as a way to instantly delete the Entry widgets' contents, a "Clear Inputs" button widget was included.

c. Methods

Adding and deleting the inputs play an important role in a task monitoring GUI. These entries will complete the functionalities of this said program.

Add Entries

This method is for the "Add" button of the program. To add the entries to the ListBox, the first process is to get the inputs of the entries using the .get() function. Then, the entry in the category ComboBox is to be inserted to the "CATEGORY" column of the ListBoxes. Next, the user entry in the Entry widget is inserted to the "TASK NAME" column of the Listboxes. And the Date selected from the calendar of the DateEntry widget is inserted to the last column "DUE DATE". After inserting the entries, the widgets are then cleared of any entries. Lastly, when an entry is blank, an error message will appear, reminding the user of any missing entry.

Delete Entries

After adding entries to the ListBox, it can then be deleted with the delete method. After selecting the category, task name, and due date in their respective ListBoxes, the selected tasks will be deleted and the empty space will be taken by the succeeding task. This method is used as a command for the "Delete" button.

Clear Inputs

The clear input method is used for the "Clear Inputs" button. It functions by using .delete() and is specified to delete the entries from the Category, Entry, and DateEntry widgets' user entries from beginning to end.

d. Title Bar

This section is used to display the title and to indicate the purpose of the created program. Since the default color of the GUI doesn't match the theme of this GUI, hence, customizing the title bar was done.

Custom Title Bar

To customize the title bar, the authors declare the overrideredirect ( ) method to remove the default title bar and change the geometry settings. Input the labelframe () function to adjust the font, colors, and buttons in the new title bar using the tk () function. To drag the program in Windows, bind the <B1-motion> key to move the application. Lastly, in order to rebuild the new title bar, create a button that closes and minimizes the application program, this will help the users to open and close the program.

Buttons

In this part the authors placed the save, load, minimize and close button, these buttons will help the users to save and retrieve the data that they inputted especially when they accidentally close this GUI.

e. Database

Saving the data of the ListBoxes is essential so that it can be restored or loaded back onto the list when the program is closed and reopened. To save the data to a single file, the authors utilized the Pickle module. The Pickle module makes it simple for the writers to store data to a .dat file and subsequently load it.

Saving

To save the tasks inserted in the list, the tasks are aggregated using .get() for each ListBox. And is then saved by creating a .dat file to contain the entries for each of the three ListBoxes. Lastly, a "Save" button was added at the title bar.

Loading

After closing the program, to reload all saved entries of the ListBoxes, the .dat files containing the tasks are opened and are inserted back to the program's ListBoxes using the "Load" button at the title bar next to the Save button.

III. Results

To review the outcome, here is the Github Link:
https://github.com/DirkTayab/OOP-1-1/tree/75d4259c7e7bd02b68bd722c8d3622d44d11eab2/WANDERBYTES%20-%20Final%20Project

This section provides the visualizations of Graphical User Interface (GUI) of the task monitoring application.

Source Code:

```
import tkinter
import tkinter as tk
from tkcalendar import DateEntry
from tkinter import *
```

```
from tkinter import messagebox
from tkinter import ttk
from PIL import ImageTk, Image
import pickle

window = Tk()
window.title("TASK MONITOR")
window.geometry('700x560')
window.config(bg="#2f2e2c")
window.resizable(FALSE,FALSE)
#window.iconbitmap("ICON.ICO")
# REMOVING TITLE BAR
window.overrideredirect(True)
#FUNCTIONS OF TITLE BAR
def move_app(e):  # runs when window is dragged
    window.geometry(f'+{e.x_root}+{e.y_root}')
def quitter(e):
    window.quit()
def mo(e):
    close_label['background'] = 'red'
def Lmo(e):
    close_label['background'] = '#2f2e2c'
def mo2(e):
    min_label['background'] = 'red'
def Lmo2(e):
   min_label['background'] = '#2f2e2c'
def min(e):
    window.withdraw()
    window.overrideredirect(FALSE)
    window.iconify()
# CREATE A NEW TITLE BAR
title_bar = Frame(window, bg="#f8de7e", relief="ridge", bd=2)
title_bar.pack(side=TOP, fill=BOTH)
# BIND TITLE BAR
title_bar.bind("<B1-Motion>", move_app)
#TITLE BAR LABEL
title_label0 = Label(title_bar, text="  ", font=("Arial", 18),bg="#f8de7e", fg="#000000")
title_label0.pack(side=LEFT, padx=2)
title_label = Label(title_bar, text="WANDERTASK:", font=("Arial", 14, "bold"),bg="#f8de7e",
fg="#000000")
title_label.pack(side=LEFT, pady=4, padx=1)
```

```
title_label2 = Label(title_bar, text="WHERE PRODUCTIVITY PEAKS", font=("Arial",
12),bg="#f8de7e", fg="#000000")
title_label2.pack(side=LEFT, pady=4, padx=3)
#TITLE BAR CLOSE
close_label = Label(title_bar, text=" x ",font=("Arial", 11), bg="#2f2e2c", fg='#f8de7e',
relief="raised")
close_label.pack(side=RIGHT,padx=4, pady=4, ipady=1 )
close_label.bind("<Button-1>", quitter)
close_label.bind("<Enter>", mo)
close_label.bind("<Leave>", Lmo)
#TITLE BAR MINIMIZE
min_label = Label(title_bar, text="-", font=("Arial", 11), bg="#2f2e2c", fg="#f8de7e",
relief="raised")
min_label.pack(side = RIGHT, padx=4, pady=4, ipadx=4, ipady=1)
min_label.bind("<Button-1>", min)
min_label.bind("<Enter>", mo2)
min_label.bind("<Leave>", Lmo2)
# LOGO
image = Image.open("WANDERTASK_LOGO.png")
# Resize the image using resize() method
resize_image = image.resize((400, 65))
img = ImageTk.PhotoImage(resize_image)
#Label for Photo
label1 = Label(image=img)
label1.image = img
label1.place(x=161, y=55)
def add_task():
    babe = cat_list.get()
    baby = task_entry.get()
    mahal = date_cal.get()
    if babe != "":
        List_task1.insert(END, babe)
        task_entry.delete(0, "end")
        if baby != "":
            List_task2.insert(END, baby)
            cat_list.delete(0, "end")
            if mahal != "":
                List_task3.insert(END, mahal)
                date_cal.delete(0, "end")
    else:
        messagebox.showwarning("warning", "Please enter some task.")
#delete btn
```

```
def del_task():
    try:
        task_index = List_task3.curselection()
        for task_index  in task_index[::-1]:
            List_task3.delete(task_index)
            task_index = List_task2.curselection()
            for task_index in task_index[::-1]:
                List_task2.delete(task_index)
            task_index = List_task1.curselection()
            for task_index in task_index[::-1]:
                List_task1.delete(task_index)
    except:
        tkinter.messagebox.showwarning(title="Warning!", message="You must select a task.")
#delete input task
def input_del():
    cat_list.delete(0,END)
    task_entry.delete(0,END)
    date_cal.delete(0,END)
#input delete color
def hold(e):
    btn_del2['background'] = 'red'
def leave(e):
    btn_del2['background'] = '#f8de7e'
#save list
def save_tasks():
    tasks1 = List_task1.get(0, List_task1.size())
    tasks2 = List_task2.get(0, List_task2.size())
    tasks3 = List_task3.get(0, List_task3.size())
    pickle.dump(tasks1, open("tasks1.dat", "wb"))
    pickle.dump(tasks2, open("tasks2.dat", "wb"))
    pickle.dump(tasks3, open("tasks3.dat", "wb"))
#Load List
def load_tasks():
    try:
        tasks1 = pickle.load(open("tasks1.dat", "rb"))
        List_task1.delete(0, tkinter.END)
        List_task2.delete(1, tkinter.END)
        List_task3.delete(2, tkinter.END)
        for task in tasks1:
            List_task1.insert(tkinter.END, task)
    except:
        tkinter.messagebox.showwarning(title="Warning!", message="Cannot find tasks.dat.")
```

```
    try:
        tasks2 = pickle.load(open("tasks2.dat", "rb"))
        List_task2.delete(0, tkinter.END)
        for task in tasks2:
            List_task2.insert(tkinter.END, task)
    except:
        tkinter.messagebox.showwarning(title="Warning!", message="Cannot find tasks.dat.")
    try:
        tasks3 = pickle.load(open("tasks3.dat", "rb"))
  List_task3.delete(0, tkinter.END)
        for task in tasks3:
            List_task3.insert(tkinter.END, task)
    except:
        tkinter.messagebox.showwarning(title="Warning!", message="Cannot find tasks.dat.")
# category list
def cat():
    cat_list["values"] = ["CPEN60",
                          "PHYS14",
                          "DCEE21",
                          "MATH12",
                          "FITT2",
                          "GNED02",
                          "GNED12", ]

cat_list = ttk.Combobox(window, values=["CPEN60",
                                        "PHYS14",
                                        "DCEE21",
                                        "MATH12",
                                        "FITT2",
                                        "GNED02",
                                        "GNED12", ], font=("Arial", 12), width=13, postcommand=cat)
cat_list.place(x=18, y=140)
lbl2 = Label(window, text = "CATEGORY ", fg="black", bg="#f8de7e", font=("Arial",
12,"bold"), width= 13)
lbl2.place(x=16,y=210)
#scrollbar to multiple listbox
def scroll(*args):
    List_task1.yview(*args)
    List_task2.yview(*args)
    List_task3.yview(*args)
#Changing Colors for buttons
def on_enter(e):
```

```
    btn_del['background'] = 'red'
def on_leave(e):
    btn_del['background'] = '#f8de7e'
def on_enter1(e):
    btn_add['background'] = 'white'
def on_leave1(e):
    btn_add['background'] = '#f8de7e'
def on_enter2(e):
    btn_save['background'] = 'white'
    btn_save['foreground'] = "#2f2e2c"
def on_leave2(e):
    btn_save['background'] = "#2f2e2c"
    btn_save['foreground'] = '#f8de7e'
def on_enter3(e):
    btn_load['background'] = 'white'
    btn_load['foreground'] = "#2f2e2c"
def on_leave3(e):
    btn_load['background'] = "#2f2e2c"
    btn_load['foreground'] = '#f8de7e'
#Entry Task
task_entry = Entry(window,font=("Arial",13), bd= 1, width=45)
task_entry.place(x=160,y=140)
lbl1 = Label(window, text = "TASK NAME ", fg="black", bg="#f8de7e", font=("Arial",
12,"bold"), width=40)
lbl1.place(x=158,y=210)
#Entry Date
lbl3 = Label(window, text = "DUE DATE ", fg="black", bg="#f8de7e", font=("Arial",
12,"bold"),width=11)
lbl3.place(x=571,y=210)
#date label
date_cal = DateEntry(window,bd=1, width=10, year=2022, month=6, day=1,justify="center",
font=12,
background='#1E4558', foreground='white', borderwidth=2)
date_cal.place(x=571, y=140)
#List box1
List_task1 = tkinter.Listbox(window, height= 19, width=
23,justify="center",exportselection="False",selectmode=MULTIPLE)
List_task1.place(x=15, y= 240)
#List box2
List_task2 = tkinter.Listbox(window, height= 19, width=
69,justify="center",exportselection="False",selectmode=MULTIPLE)
List_task2.place(x=154, y= 240)
```

```
#List box3
List_task3 = tkinter.Listbox(window, height= 19, width=
19,justify="center",exportselection="False",selectmode=MULTIPLE)
List_task3.place(x=568, y= 240)
#scroll settings
scrollbary = tk.Scrollbar(window,orient=tk.VERTICAL, command=scroll)
List_task1.config(yscrollcommand=scrollbary.set)
List_task2.config(yscrollcommand=scrollbary.set)
List_task3.config(yscrollcommand=scrollbary.set)
#scrollbary.pack(side=tk.RIGHT, fill=tk.Y)
#scrollbary.grid(row=1, column=5, sticky=tk.NS)
scrollbary.place(x=670, y=240,height=308, width=17)
#load button
btn_load = tkinter.Button(title_bar, text="Load",font=("Arial", 8), width=4,bg="#2f2e2c",
fg="#f8de7e",command=load_tasks)
btn_load.bind("<Enter>", on_enter3)
btn_load.bind("<Leave>", on_leave3)
btn_load.place(x=590,y=6)
#save button
btn_save = tkinter.Button(title_bar, text="Save", font=("Arial", 8), width=4,bg="#2f2e2c",
fg="#f8de7e",command=save_tasks)
btn_save.bind("<Enter>", on_enter2)
btn_save.bind("<Leave>", on_leave2)
btn_save.place(x=550,y=6)
#Add button
btn_add = tkinter.Button(window, text="Add", width=5, command=add_task,bg="#f8de7e")
btn_add.place(x=20, y=170)
btn_add.bind("<Enter>", on_enter1)
btn_add.bind("<Leave>", on_leave1)
#delete input task
btn_del2 = tkinter.Button(window, text="Clear Inputs",
width=9,command=input_del,bg="#f8de7e")
btn_del2.place(x=611, y=170)
btn_del2.bind("<Enter>", hold)
btn_del2.bind("<Leave>", leave)
#Delete Button
btn_del = tkinter.Button(window, text="Delete", width=5,command=del_task,bg="#f8de7e")
btn_del.place(x=69, y=170)
btn_del.bind("<Enter>", on_enter)
btn_del.bind("<Leave>", on_leave)

window.mainloop()
```

Output

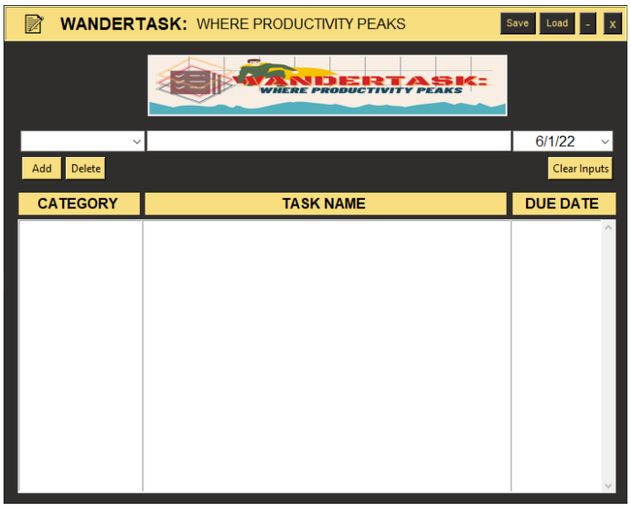

Figure 5. Final Output

This task monitoring allows the users to create tasks and track the progress through their various stages from start up to when they finish them. It performs basic functions such as allowing the users to select a category, input the title of the given tasks and its deadline. It also allows the user to clear all inputs at once and delete an entry from the listbox if the inputted data changes or if the task is already finished. The authors also included the GUI's name and logo to make it more appealing to the user's eye. Aside from that, the name and logo will make it easy for the user to recognize the goal of the GUI as well as the authors who created it. Finally, the writers included buttons to store and load data. Regardless of whether the user closes the software and tries to repeat it, the data entered will be saved and retrievable.

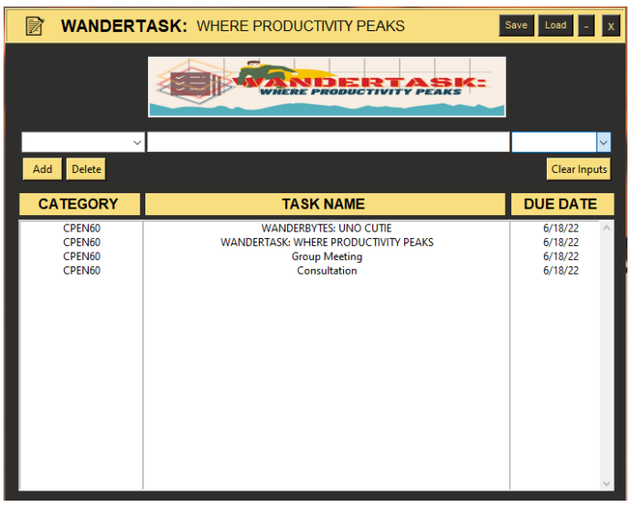

Figure 6. Adding a task

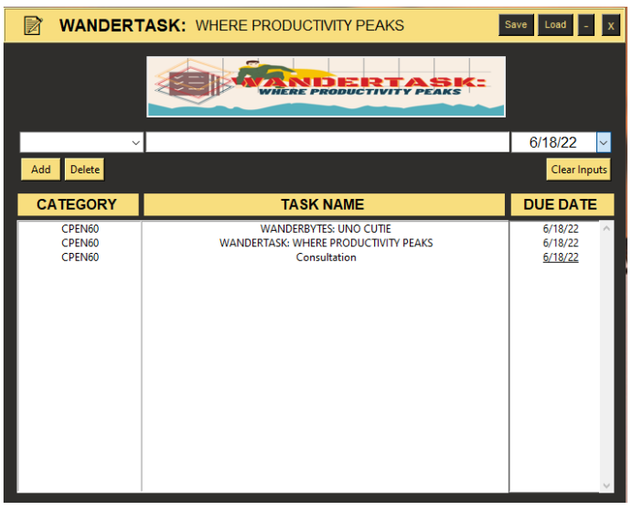

Figure 7. Deleting a Task

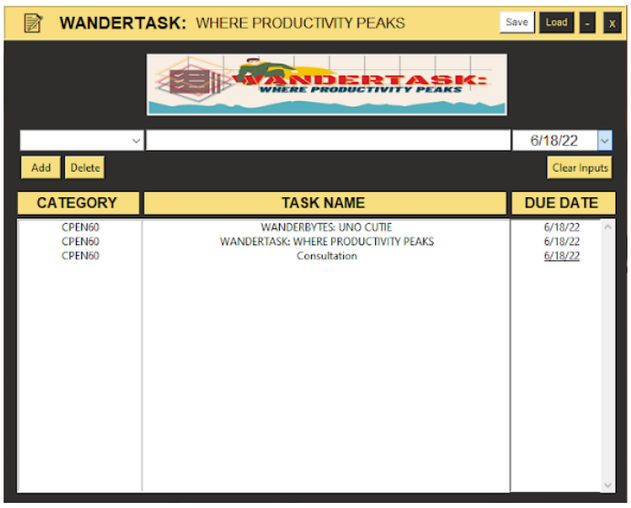

Figure 8. Saving data

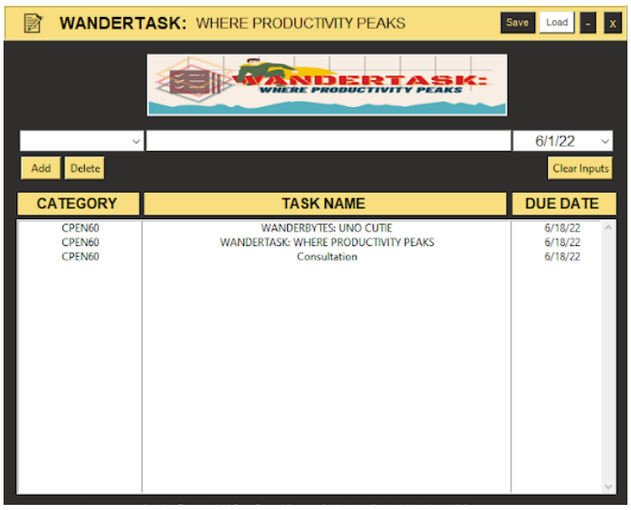

Figure 9. Loading data after reopening the GUI

IV. Conclusion

In this project, the authors were asked to create a program that would serve as their final output. The authors considered every bit of knowledge they got through their subject, and came up with an output named Wandertask: Where Productivity Peaks. To be specific, this program is similar to a simple task monitoring application.

Task Monitoring is the process of managing an activity with a fixed start and end date through the stages of planning, execution, monitoring, and closing. Its goal is to achieve a specific end goal by performing several manageable tasks. The authors developed a program for individuals, particularly students, to assist them in monitoring their time spent on tasks, ongoing and completed tasks, workload, and performance. This newly created program is easy to use and provides task monitoring GUI that anyone can start using right away. It helps individuals effectively in scheduling, managing, and organizing time and activities. In addition, it can be used to balance workloads, predict bottlenecks, and avoid delays and missed deadlines.

In creating this project, the authors imported some features from the Tkinter, Tkcalendar, Pickle and Pillow (PIL) modules that would be a big help in achieving their desired outcome and functionalities. They also conducted research from different reliable online platforms for them to successfully create an effective task monitoring application. As they continue to build the said program, importing some features causes different errors and problems, such as using unfamiliar codes and putting a timer as one of the authors' desired features. Moreover, the collaboration and determination by the authors continued and resulted in innovating new methods and codes to successfully run the program.

Throughout this final project, the writers were able to utilize all of their Object-Oriented Programming skills to create this GUI. Despite all of the difficulties and several errors, they ultimately believed in themselves to continue their effort. The authors briefly presented the methodology of how they constructed the task monitoring from the start to the completion. It is currently a fully established initiative that seeks to assist individuals at their optimum productivity.

Reference

Website

B. Kumar, “Python tkinter todo list (build step by step),” Python Guides, 26-Mar-2021. [Online]. Available: https://pythonguides.com/python-tkinter-todo-list/. [Accessed: 09-Jun-2022].

B. Oakley , “Listbox with checkbuttons in python with Tkinter,” Stack Overflow, 01-Oct-1966. [Online]. Available: https://stackoverflow.com/questions/54770537/listbox-with-checkbuttons-in-python-with-tkinter. [Accessed: 08-Jun-2022].

“How to resize image in python - tkinter?,” GeeksforGeeks, 31-Aug-2021. [Online]. Available: https://www.geeksforgeeks.org/how-to-resize-image-in-python-tkinter/. [Accessed: 12-Jun-2022].

“Python tkinter tutorial: Simple to do list app,” YouTube, 24-Oct-2020. [Online]. Available: https://youtu.be/8qUJ9a_3zSQ. [Accessed: 07-Jun-2022].

“Select all in a Tkinter Listbox,” Stack Overflow, 01-May-1961. [Online]. Available: https://stackoverflow.com/questions/19196130/select-all-in-a-tkinter-listbox. [Accessed: 11-Jun-2022].

Techopedia, “What is task management?” Techopedia.com, 02-Jun-2015. [Online]. Available: https://www.techopedia.com/definition/9652/task-management. [Accessed: 12-Jun-2022].

“Tkinter hovering over button -> color change,” Stack Overflow, 01-Nov-1965. [Online]. Available: https://stackoverflow.com/questions/49888623/tkinter-hovering-over-button-color-change. [Accessed: 10-Jun-2022].

“Using Icons, Images, and Exit Buttons - Python Tkinter GUI Tutorial #8,” YouTube, 29-Jan-2019. [Online]. Available: https://youtu.be/NoTM8JciWaQ. [Accessed: 11-Jun-2022].

Zoho, “What is task management?” Zoho. [Online]. Available: https://www.zoho.com/projects/task-management.html. [Accessed: 11-Jun-2022].