Task Management System

Introduction to Programming

CMPT 120L

Team Purple

Marist College

School of Computer Science and Mathematics

Submitted To:

Dr. Reza Sadeghi

Fall 2022

Project Progress Report 04 of Task Management System

Team Name

Team Purple

Team Members

1. Abel Scholl Anna.Scholl1@marist.edu (Team Head)

I am a freshman from Wilkes-Barre, Pennsylvania, and am majoring in Computer Science. I have taken four programming courses previously and have learned Python and Java. My interests include art, animation, video games, crocheting, sewing, and general creation of things. I like programming, because there is always something tangible that gets created that is interactive, personalized, and serves a purpose. I chose my group based on people in my class I thought seemed nice and wanted to connect to more. I took on the role as team head, because I have the most programming experience. I am excited to work with everyone, and I think we will work well together.

2. Christelle Bernardin Christelle.Bernardin1@marist.edu (Team Member)

I'm Christelle, I like it pronounced Crystal, and I'm a freshman at this lovely college. This is my first time taking a Computer Science class. I like listening to music, sleeping, sesame chicken, and I really want to become the best student I can be. I haven't known my group members for long, but I already know we're going to work successfully together. I get an easy-going vibe from them, which I really like. It was clear the team head would be Abel, as they have had experience with GitHub before, and I don't doubt they are an excellent student.

3. Sydney George Sydney.George1@marist.edu (Team Member)

Hello, I'm Sydney! Some of my favorite things include reading, art, plants, and tennis. I'm a freshman, so I don't have any programming experience, but I am dedicated to learning and excited about this project. I wanted to work with my team members because they seem like they are team players who are willing to solve issues together and are good communicators. In the brief times we have talked, everyone seems personable and flexible, so I think we will work well together. We selected Abel as our team head because the rest of us weren't as confident using GitHub, and I think having someone with more experience/knowledge as our leader is wise.

CMPT 120L Project_Phase_04_TeamPurple

Table of Contents

Table of Figures

Project Description

Objective and Module Descriptions:

The purpose of the Task Management System project is to create a graphical user interface using python that is an interactive calendar for a logged in user to create, edit, and organize tasks on their schedule. It will include administrative and normal user logins that have different permissions available for the user to edit the interface. It will have a login window that requests a username and password, allowing the user to log in to the interface, add an account, and remove accounts, a welcome page where a user can request to view their calendar by day, week, month, or year, as well as edit, add or remove tasks, and search. It will also include an exit feature to log out of the interface.

GitHub Repository Address

https://github.com/Abel-Scholl/CMPT102L_TaskManagementSystem_TeamPurple#cmpt102l_taskmanagementsystem_teampurple

Graphical User Experience Design

Complete Flowchart

A user's experience will begin with the login page. If the user is authroized, they will be directed to the main page. From the main page, the user can select from the button options of settings, add, edit, and remove. Settings will change login information, add will require the user to input information for a new task, edit allow the user to change that information, and remove will delete a task. The user can also use the month/year comboboxes to update the calender and select a day to output task information for. The user can use the search combobox to select a criteria they want to search by then press go to search. At any point in the program/flowchart, the user can select the exit button and the current window in use will close.

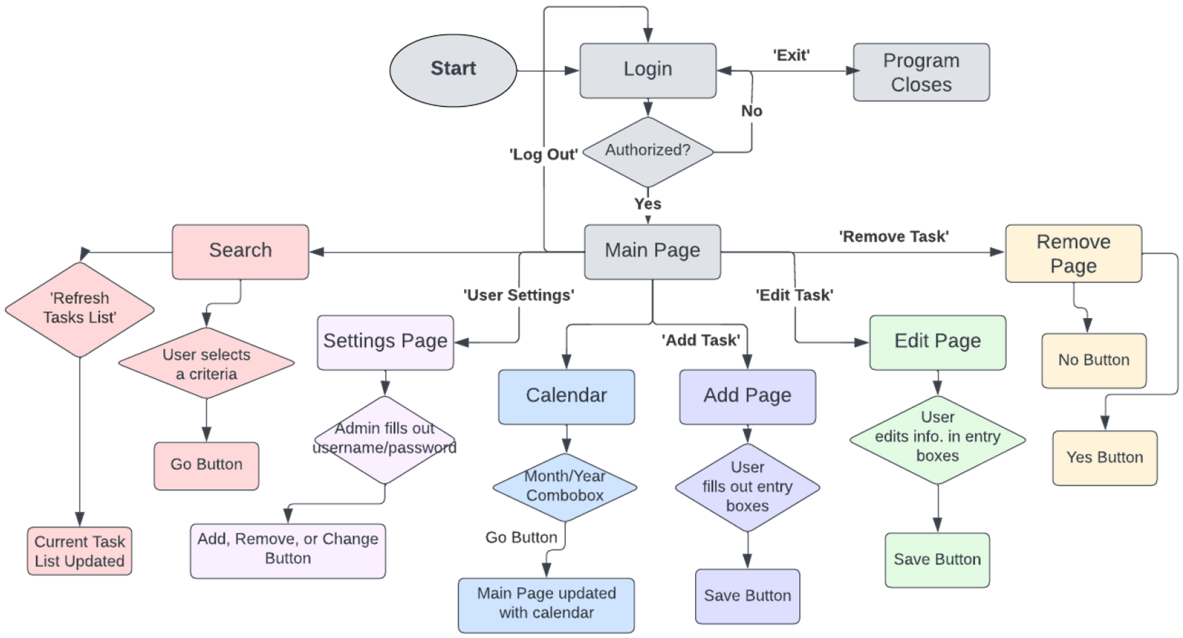


Figure 1: Complete Flowchart (Lucidchart)

Graphical User Experience Design

Login Page

This page functions by requiring the user to input text corresponding to a username and password in entry boxes. A valid username and password will grant a user access to the main page of the task management system. An invalid username or password will output a warning statement and require the user to retry.

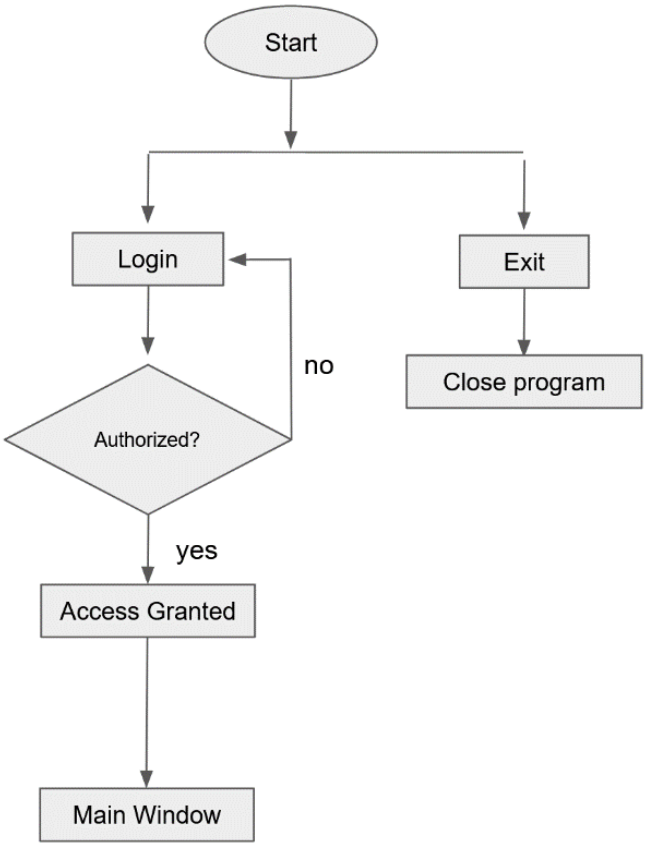


Figure 2: Login Page Flowchart (Lucidchart)

Graphical User Experience Design

Main Window

The main window allows the user to view their calendar and saved tasks. The user can cycle through comboboxes of months and years that change the layout of the calendar as necessary for that specific month and year combination. Upon opening the window, the user can also view their tasks for the current day. By selecting a day on the calendar, the tasks for that day will be visible. The user can also select a task and press buttons to edit or remove the selected task. The user can add a new task by pressing the “Add” button which will open the Add window, search for a task by pressing the ”search” button which will open the Search window, and go to the Settings window through the “Settings” button.

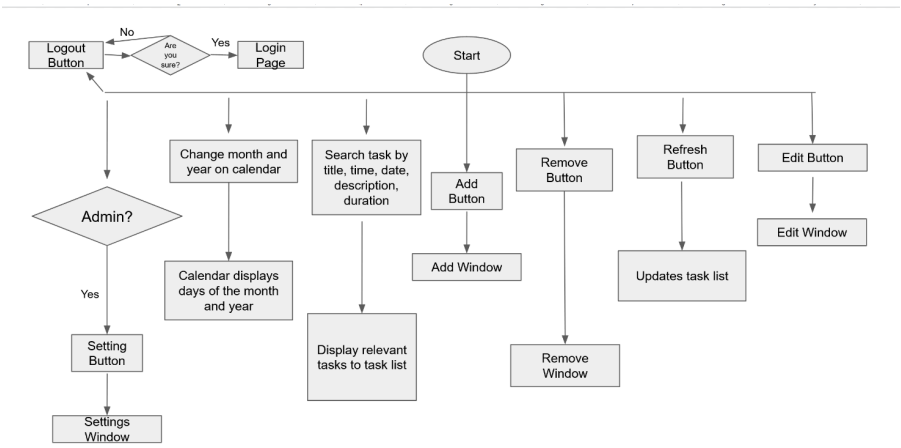


Figure 3: Main Page Flowchart (Lucidchart)

Graphical User Experience Design

Search Function

The search function on the Main Page allows users to search for a task(s) by using the categories of title, time, date, description and duration in a drop-down menu. The user will have to pick a selection from the drop-down and enter text before pressing the search button in order for the input to be valid. This function will output a list of any tasks that match the inputted information.

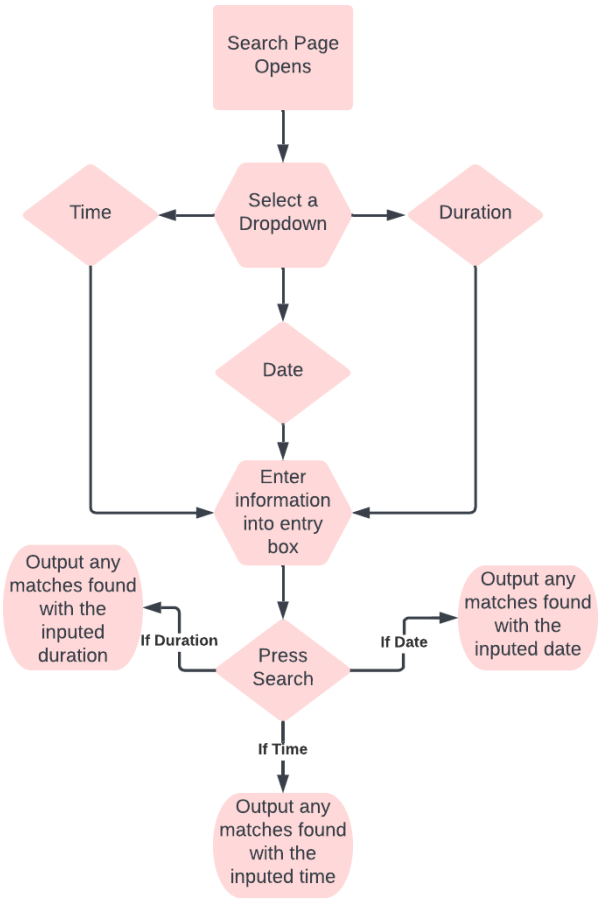


Figure 4: Search Function Flowchart (Lucidchart)

Graphical User Experience Design

Add Page

This page allows the user to add a task by manually inputting the title, duration, time, and description of the task. The user will have to click a save button for the task to save. There is no physical output, instead, the task data gets stored for future reference. Or, the user can press (input) an exit button and the window will close (output).

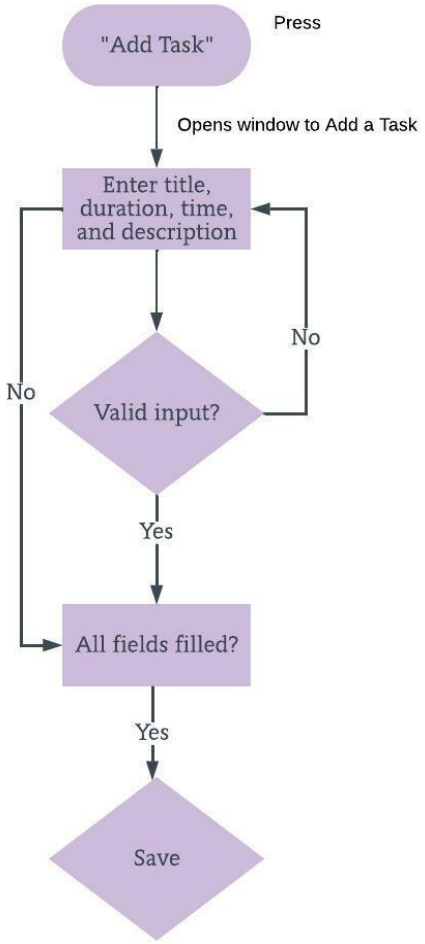


Figure 5: Add Page Flowchart (Lucidchart)

Graphical User Experience Design

Edit Page

This page allows a user to change their previous inputs of title, duration, time, and description for a task by altering the characters in the input fields. The user must hit a save edit button for the input changes to save. There is no physical output, instead, the updated task data gets stored for future reference. Or, the user can press (input) an exit button and the window will close (output).

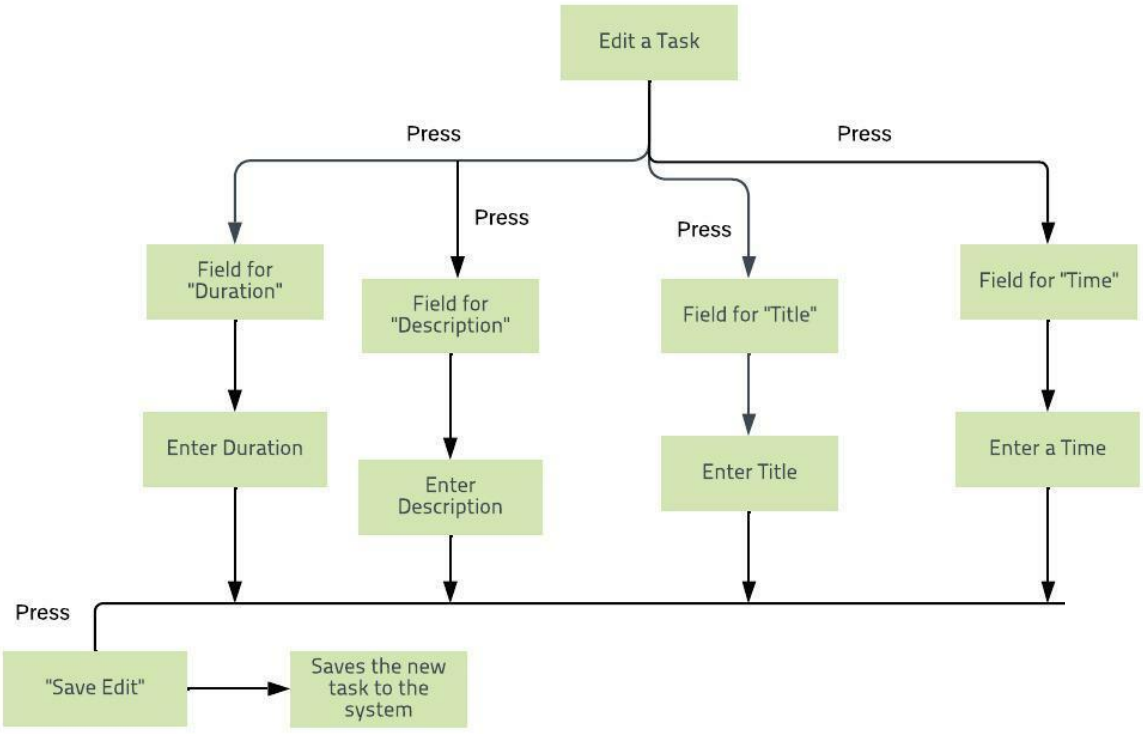


Figure 6: Edit Page Flowchart (Lucidchart)

Graphical User Experience Design

Remove Page

The function of this page is to remove a task. The input will be a button press, yes or no. The output will either be the task is deleted or the task remains. After both output possibilities, the window will close.

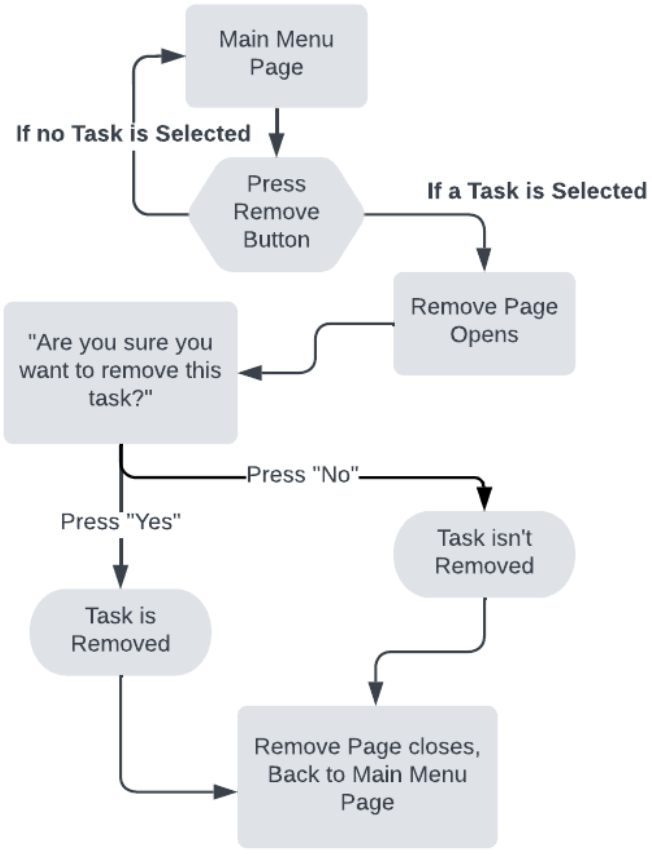


Figure 7: Remove Page Flowchart (Lucidchart)

Graphical User Experience Design

Settings Page

This page requires users to input a username and password. The user must then provide a secondary input which is a button press from the options of ‘add user’, ‘remove user’, and ‘change admin details’. If the username/password is valid, the output will vary based on the button input. The inputted username/password will either be entered into the system, removed, or changed (if admin).

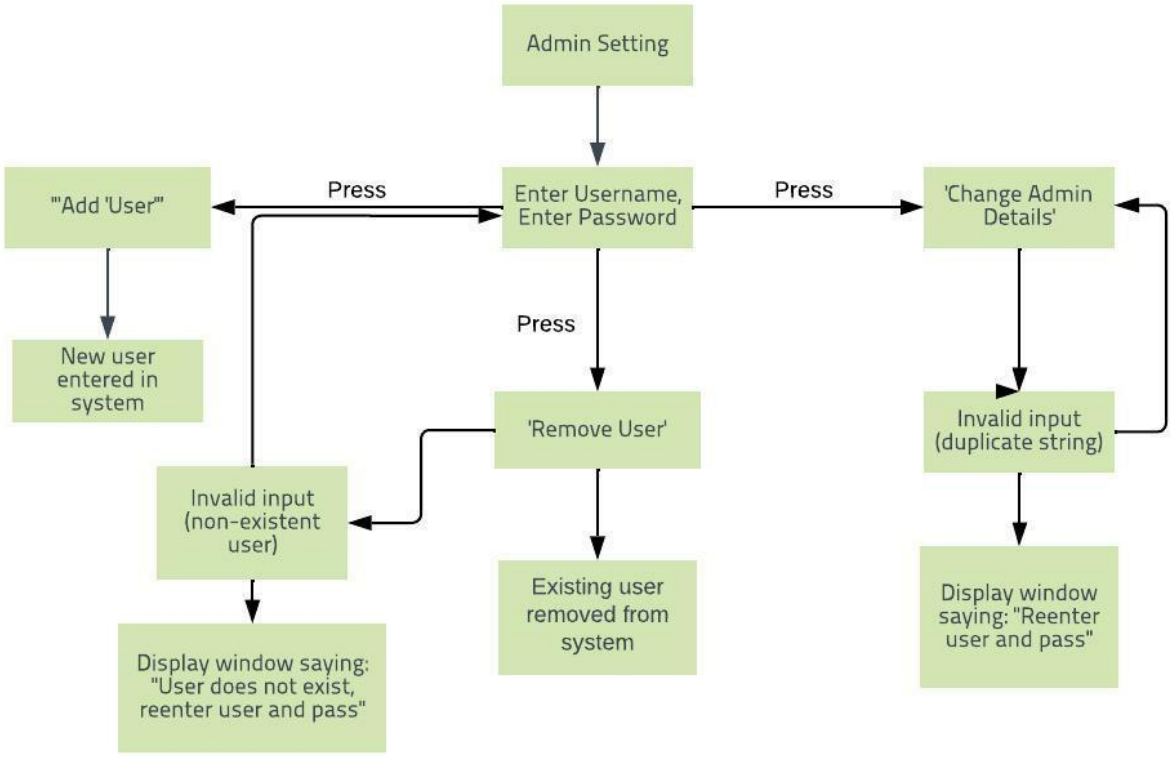


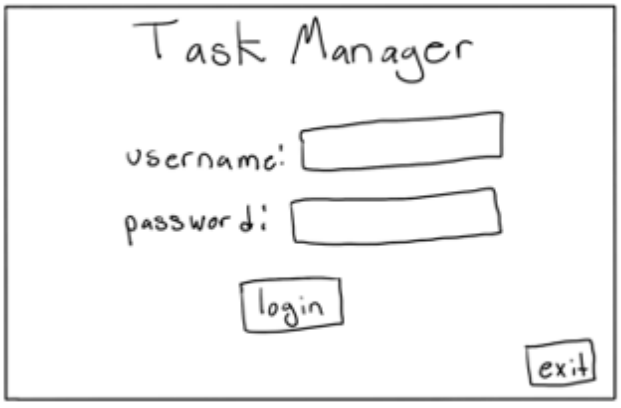
Figure 8: Settings Page Flowchart (Lucidchart)

Graphical User Interface Design

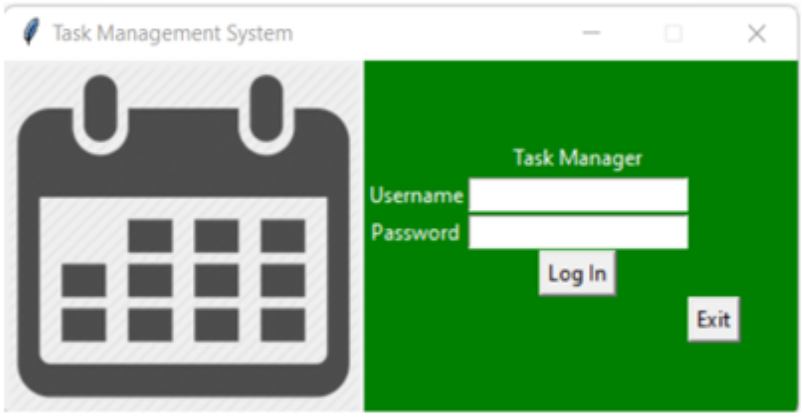
Login Page

The design idea for the login page is to have it remain simple and user-friendly. The title of the program will be displayed at the top to identify it to the user, and the page will contain a simple entry box for both the username and password along with a login and exit button. Our implementation followed this design and used the color green as a background so it wasn't plain. The implementation also includes an image.

Design Idea:



Implementation of Design:



Figures 9 & 10: Login Page Layout Designs

Graphical User Interface Design

Login Page

Python Code for the Implementation:

```
##TMS Login
##11/19/22 Updates
## login system works as long as users are saved in csv
## connected to main page


import tkinter as tk
from tkinter import messagebox
from tkinter import *
import Class as TC #name for the file of User and task classes
from PIL import Image, ImageTk # import for image
import csv
import Main as M # importing main file of main page


### Only need to run this once to get admin then can comment it
##ADMIN = TC.User("A","6", True)##initialize admin w/username and password
##ADMIN.createNewUser() ##this line saves the new user to the csv


def log(username, password, window):
    # checks the csv file to see if inputted username
    # and password matches saved csv file
    # if yes, continues to main window
    # else, error message
    with open ('userLoginInfo.csv', 'r') as file:
        csvReader = csv.reader(file)
        found=False
        for member in csvReader:
            if member[0]==username and member[1]==password:
                found = True
                messagebox.showinfo('Access Granted', f'Welcome, {member[0]}!')
                user=member
                ##save the current user to a text file
                with open ('CurrentUser.txt', 'w') as f:
                    f.write(str(member[0]))
                f.close()
                Exit(window)
                M.mainFunc(member)
            else: continue
        if not found:messagebox.showwarning('Access Denied', 'User not found.')

    # Storing User Entry for later in add
    with open ('CurrentUser.csv', 'w', newline = '') as file:
        csvWriter = csv.writer(file) # 'W' mode replaces all, so the csv doesn't have to be later deleted
        csvWriter.writerow([username])


def Exit(window):
    # closes the window when exit button is clicked
    window.destroy()
```

```
def Exit(window):
    # closes the window when exit button is clicked
    window.destroy()

def LogPage():

    BG_COLOR = "Green"
    FG_COLOR = "White"

    # Create the Window
    window=tk.Tk()
    window.title('Task Management System')
    window.geometry('460x200')
    window.configure(bg = 'Green')
    # Create the Frames
    BodyFrame=tk.Frame(window, bg=BG_COLOR, height=300, width=200)
    BodyFrame.grid(row=1, column = 2, padx=10)

    PicFrame = tk.Frame(window, bg = BG_COLOR, height = 300, width = 200)
    PicFrame.grid(row = 1, column = 1)

    ##window.resizable(width = False, height = False)

    # Labels
    titleLabel = tk.Label(BodyFrame, text="Task Manager", bg = BG_COLOR, fg= FG_COLOR, font=("Comic Sans MS", 14))
    titleLabel.grid(column=2,row=0)

    userLabel = tk.Label(BodyFrame, text="Username", bg = BG_COLOR, fg= FG_COLOR, font=("Comic Sans MS", 11))
    userLabel.grid(column=1,row=1)

    passLabel = tk.Label(BodyFrame, text="Password", bg = BG_COLOR, fg= FG_COLOR, font=("Comic Sans MS", 11))
    passLabel.grid(column=1,row=2)

    # Image
    img = Image.open("LogPage.png") # load image
    resized_image = img.resize((200,200), Image.Resampling.LANCZOS) # resize, remove structural padding
    new_image = ImageTk.PhotoImage(resized_image)# convert to photoimage
    label = Label(PicFrame, image = new_image) #display the image
    label.grid(row=2, column = 4)


    ##username and password entrys
    userEntry = tk.Entry(BodyFrame)
    userEntry.grid(column=2,row=1)

    passEntry = tk.Entry(BodyFrame)
    passEntry.grid(column=2,row=2)

    loginButton = tk.Button(BodyFrame, text='Log In', command = lambda:log(userEntry.get(), passEntry.get(), window))
    loginButton.grid(column=2,row=5)

    exitButton=tk.Button(BodyFrame, text='Exit', command=lambda:Exit(window))
```

```
    ##username and password entrys
    userEntry = tk.Entry(BodyFrame)
    userEntry.grid(column=2,row=1)

    passEntry = tk.Entry(BodyFrame)
    passEntry.grid(column=2,row=2)

    loginButton = tk.Button(BodyFrame, text='Log In', command = lambda:log(userEntry.get(), passEntry.get(), window))
    loginButton.grid(column=2,row=5)

    exitButton=tk.Button(BodyFrame, text='Exit', command=lambda:Exit(window))
    exitButton.grid(column=3, row=6)


    window.mainloop()


if __name__ == "__main__":
    LogPage()
```

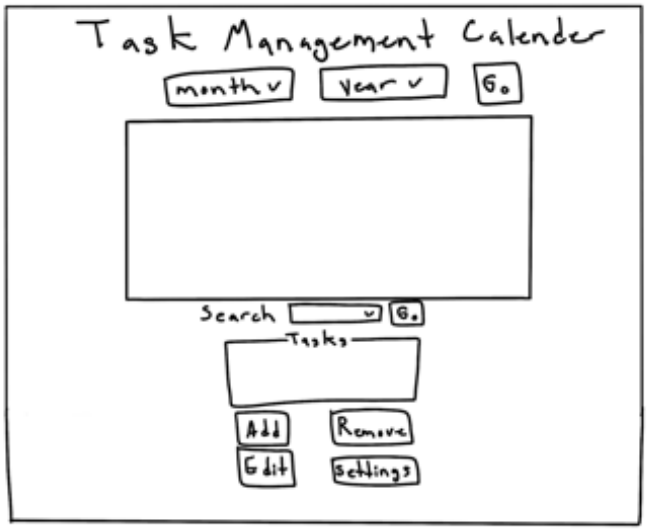
Figure 11: Login Page Code Implementation

Graphical User Interface Design

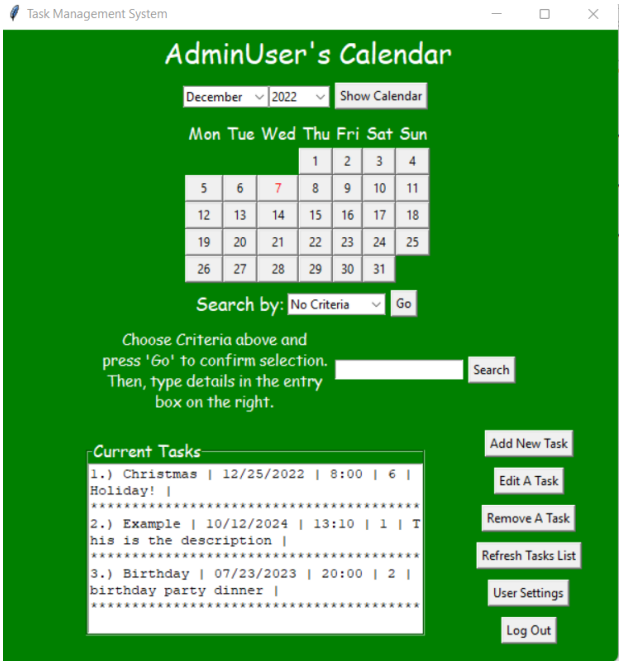
Main Window

The main window will contain the most interactive elements of all the interface designs. The title will be displayed at the top, followed by drop-down boxes for month and year. There will be a big open space for the calendar to appear. The design also includes a search function, output area, and other buttons. For the implementation, we did not include any images in order to not clutter the interface. There is also a difference in that the title was changed to be more personal for each user, and the buttons are in a different arrangement.

Design Idea:



Design Implementation:



Figures 12 & 13: Main Page Layout Designs

Graphical User Interface Design

Main Window

Python Code for the Design Implementation:

```
'''
Main Window
'''
import calendar
import datetime
from tkinter.ttk import Combobox
from tkinter import *
import tkinter as tk
from tkinter import messagebox
import Class as TC # import class file
from PIL import Image, ImageTk
import csv
import Login

##Main page of the Task Management System.
#Includes interactive calendar
#Houses links to all other functions
#Includes search function
##displays tasks in a text box

TASKFILE = "taskList.csv"

def logOut(window, user):
    if messagebox.askyesno('Log Out', 'Are you sure you want to log out?'):
        window.destroy()
        Login.LogPage()


def printTasksToTextBox(user,tasksTextBox):
    tasksTextBox.delete(1.0,END)
    num=1
    with open(TASKFILE, 'r') as file:
        csvReader=csv.reader(file)
        for member in csvReader:
            if member[0] == user[0]:
                string = f'{num}.) '
                for element in member: ##prints task details excluding username
                    if element != member[0]:
                        string = string+element+" | "
                string=string+"\n****************************************"
                tasksTextBox.insert(END, string)
                num+=1


def searchFunc(criteria, tasksTextBox, user, details):
    '''this function will display tasks
    to the Task List text box on the main page'''
    tasksTextBox.delete(1.0,END)
    index = 0
    match criteria:
        case "Title": index=1
        case "Date": index=2
```

```
def searchFunc(criteria, tasksTextBox, user, details):
    '''this function will display tasks
    to the Task List text box on the main page'''
    tasksTextBox.delete(1.0,END)
    index = 0
    match criteria:
        case "Title": index=1
        case "Date": index=2
        case "Time": index=3
        case "Duration": index=4
        case "Description": index=5

    ##open the CSV file in read mode
    num=1
    with open(TASKFILE, 'r') as file:
        csvReader=csv.reader(file)
        for member in csvReader:
            if member[0] == user[0]:
                if member[index] == details:
                    string = f'{num}.) '
                    for element in member:
                        if element != member[0]:
                            string = string+element+" | "
                    string=string+"\n**************************************"
                    tasksTextBox.insert(END, string)
                    num+=1



def searchByResponse(sF, criteria, tasksTextBox, user):
    ##sF = searchFrame2
    ##criteria = retult of search combobox
    ##details is what was written in the entry
    widget_list = all_children(sF)
    for item in widget_list:
        item.destroy() ##destroys each widget in frame to be replaced later

    t=""
    match criteria:
        case "Date":
            t="Enter a Date (format: mm/dd/yyyy):"
        case "Time":
            t="Enter a Time (military time, use colon):"
        case "Title":
            t="Enter Task Title:"
        case "Duration":
            t="Enter Duration (hours): "
        case "Description":
            t="Enter Task Description: "
        case "No Criteria":
            t ='''Choose Criteria above and
```

```
def responsiveCalendar(window, user, yC, monthsList, mC, cF, tasksTextBox):
    widget_list = all_children(cF)
    for item in widget_list:
        item.destroy() ##destroys each widget in frame
    m=monthsList.index(mC)+1
    today = datetime.date.today()
    dayNum=0
    x = calendar.monthrange(int(yC), m)
    daysInMonth = x[1]
    k=(datetime.date(int(yC), m, 1)).weekday() ##the first day of the chosen month of chosen year
    count = 0

    ##Calendar frame and days of the week labels
    days = ["Mon", "Tue","Wed","Thu","Fri","Sat", "Sun"]
    dateToday = datetime.date.today()
    ##creates a label for each weekday 0-6 starting with monday (0)
    for d in range(len(days)):
        weekdayLabel = Label(cF, text=days[d], bg='green', fg='white', font=("Comic Sans MS", 12))
        weekdayLabel.grid(column=d, row=0)


    ##Creates a button for each day and displays as a calendar
    for i in range(6):
        for j in range(7):

            if dayNum < daysInMonth:
                if count<k: ##creates a label for as many days before first day of month
                    label= Label(cF, text="   ", bg='green')
                    label.grid(row=i+1, column=j, sticky='nsew')
                    count+=1
                else:
                    dayNum+=1
                    if today.day == dayNum and today.month == m and str(today.year)==yC:
                        button = Button(cF, text=f"{dayNum}", fg="red")
                        button.grid(row=i+1, column=j, sticky='nsew')
                    else:
                        button = Button(cF, text=f"{dayNum}")
                        button.grid(row=i+1, column=j, sticky='nsew')
    cF.pack(side=TOP)



def mainFunc(user):
    window = Tk()

    window.title('Task Management System')
    window.geometry('600x610')
    window.configure(bg='green')

    BG_COLOR = "Green"
    FG_COLOR = "White"
```

```
    t=...
    match criteria:
        case "Date":
            t="Enter a Date (format: mm/dd/yyyy):"
        case "Time":
            t="Enter a Time (military time, use colon):"
        case "Title":
            t="Enter Task Title:"
        case "Duration":
            t="Enter Duration (hours): "
        case "Description":
            t="Enter Task Description: "
        case "No Criteria":
            t ='''Choose Criteria above and
press 'Go' to confirm selection.
Then, type details in the entry
box on the right.'''

    searchLabel2=Label(sF, text=t, bg="Green", fg='white', font=("Comic Sans MS", 11))
    searchLabel2.pack(side=LEFT)


    searchEntry = Entry(sF)
    searchButton=Button(sF, text="Search", command=lambda:searchFunc(criteria, tasksTextBox, user, searchEntry.get()))

    searchButton.pack(side=RIGHT)
    searchEntry.pack(side=RIGHT, padx=5)




def all_children(frame):
    ##this function references all widgets in a frame
    ##and returns it as a list
    ##made to be recursive so another function
    ##can iterate through and destory all widgets but not the entire frame
    _list = frame.winfo_children() ##list of cF's widgets from bottom to top
    for widget in _list:
        if widget.winfo_children():
            _list.extend(window.winfo_children())

    return _list


def responsiveCalendar(window, user, yC, monthsList, mC, cF, tasksTextBox):
    widget_list = all_children(cF)
    for item in widget_list:
        item.destroy() ##destroys each widget in frame
    m=monthsList.index(mC)+1
    today = datetime.date.today()
    dayNum=0
```

```
BG_COLOR = "Green"
FG_COLOR = "White"

##Welcome Banner
welcFrame = Frame(master=window)
welcomeBanner = Label(welcFrame, text=(user[0]+"'s Calendar"),
                      bg='green', fg='white', font = ("Comic Sans MS",20))
welcomeBanner.pack(side=LEFT)
welcFrame.pack()


##Months and Years as combo boxes + go button
dateFrame = Frame(master=window, bg='green')

monthsList = ["January", "February", "March", "April",
                  "May", "June", "July", "August",
                  "September","October","November","December"]

monthsCombo = Combobox(dateFrame,width=10,state="readonly")
monthsCombo['values']= (monthsList)
monthsCombo.current((datetime.date.today().month)-1)
monthsCombo.pack(side=LEFT)


yearsList = ["2022", "2023", "2024", "2025",
                  "2026", "2027", "2028", "2029",
                  "2030","2031","2032","2033"]

yearsCombo = Combobox(dateFrame,width=6,state="readonly")
yearsCombo['values']= (yearsList)
yearsCombo.current((datetime.date.today().year)-2022)
yearsCombo.pack(side=LEFT, pady=10)


calendarFrame=Frame(master=window, bg="Green")
showCalendarButton = Button(dateFrame, text="Show Calendar",
                    command=lambda:responsiveCalendar(window, user,
                   yearsCombo.get(), monthsList, monthsCombo.get(), calendarFrame, tasksTextBox))
showCalendarButton.pack(side=LEFT, padx=5)

dateFrame.pack()

##Frame for task related options
##including task list,
##task options (add, edit, remove)

taskFrame = Frame(master=window, bg= 'green')
taskLabelFrame = LabelFrame(taskFrame, text='Current Tasks', bg="Green", fg='white', font = ("Comic Sans MS",12))
```

```
    ##if tasks exist for that day. else, label says: "There are no tasks scheduled for this day"
    tasksTextBox = Text(taskLabelFrame, width = 40, height=10)
    tasksTextBox.pack(side=BOTTOM)
    taskLabelFrame.pack(side=LEFT,padx=50)

    ##call responsive calendar function to display calendar
    responsiveCalendar(window, user, yearsCombo.get(), monthsList, monthsCombo.get(), calendarFrame, tasksTextBox)


    # Buttons
    import Add
    import Remove
    import Edit
    import Settings

    addButton = Button(taskFrame, text="Add New Task", command=lambda:Add.AddFunc(window=Toplevel()))
    editButton = Button(taskFrame, text="Edit A Task", command=lambda:Edit.EditFunc(window=Toplevel()))
    removeButton = Button(taskFrame, text="Remove A Task", command=lambda:Remove.RemoveWin(window=Toplevel))
    refreshButton = Button(taskFrame, text="Refresh Tasks List", command = lambda:printTasksToTextBox(user,tasksTextBox))
    addButton.pack(pady=5)
    editButton.pack(pady=5)
    removeButton.pack(pady=5)
    refreshButton.pack(pady=5)

    settingsButton=Button(taskFrame, text="User Settings", command=lambda:Settings.Set(window=Toplevel()))
    if user[-1]=='TRUE': ##if admin is True? It's supposed to be if false but it only works backwards so I'm leaving it like this.
        settingsButton.pack(pady=5)

    logoutButton=Button(taskFrame, text="Log Out", command=lambda:logOut(window, user))
    logoutButton.pack(pady=5)

    ##Frame for search function
    searchFrame1 = Frame(master=window, bg="Green")
    searchLabel1=Label(searchFrame1, text="Search by:", bg="Green", fg='white', font=("Comic Sans MS", 13))
    searchLabel1.pack(side=LEFT)

    searchByCombo = Combobox(searchFrame1, width=12, state="readonly")
    searchByCombo['values']= ("No Criteria", "Date", "Time",
                      "Title", "Duration", "Description")
    searchByCombo.set("No Criteria Set")
    searchByCombo.current(0)

    searchFrame2 = Frame(master=window, bg="Green")
    searchByResponse(searchFrame2, searchByCombo.get(), tasksTextBox, user)
    goButtonSearch = Button(searchFrame1, text="Go",
                            command=lambda:searchByResponse(searchFrame2, searchByCombo.get(), tasksTextBox, user))

    searchByCombo.pack(side=LEFT, pady=10)
    goButtonSearch.pack(side=LEFT, padx=5)
```

```
    ##Frame for search function
    searchFrame1 = Frame(master=window, bg="Green")
    searchLabel1=Label(searchFrame1, text="Search by:", bg="Green", fg='white', font=("Comic Sans MS", 13))
    searchLabel1.pack(side=LEFT)

    searchByCombo = Combobox(searchFrame1, width=12, state="readonly")
    searchByCombo['values']= ("No Criteria", "Date", "Time",
                      "Title", "Duration", "Description")
    searchByCombo.set("No Criteria Set")
    searchByCombo.current(0)

    searchFrame2 = Frame(master=window, bg="Green")
    searchByResponse(searchFrame2, searchByCombo.get(), tasksTextBox, user)
    goButtonSearch = Button(searchFrame1, text="Go",
                            command=lambda:searchByResponse(searchFrame2, searchByCombo.get(), tasksTextBox, user))

    searchByCombo.pack(side=LEFT, pady=10)
    goButtonSearch.pack(side=LEFT, padx=5)


    searchFrame1.pack(side=TOP)
    searchFrame2.pack(side=TOP)
    taskFrame.pack(side=TOP, pady=10)

    printTasksToTextBox(user,tasksTextBox)

    window.mainloop()

#mainFunc(['A'])
```

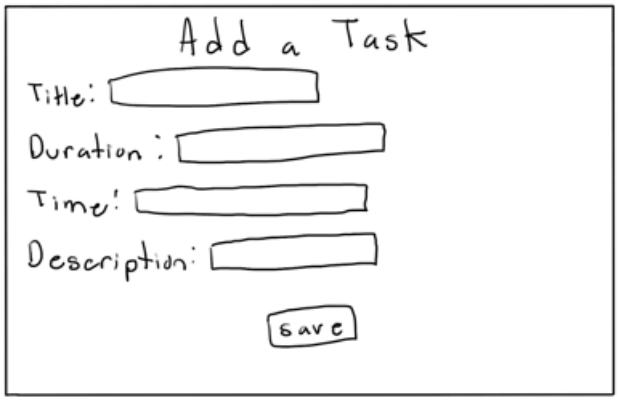
Figure 14: Main Page Code Implementation

Graphical User Interface Design

Add Page

The add page design includes a descriptive title, “Add a Task”, at the top indicating the page. There will be entry boxes labeled according to their purpose of title, duration, time, or description. In our design, the entries will be followed by save and exit buttons. The ultimate implementation includes an additional entry field of data as well as an image in the banner at the top. The background is purple.

Design Idea:



Implementation of Design:

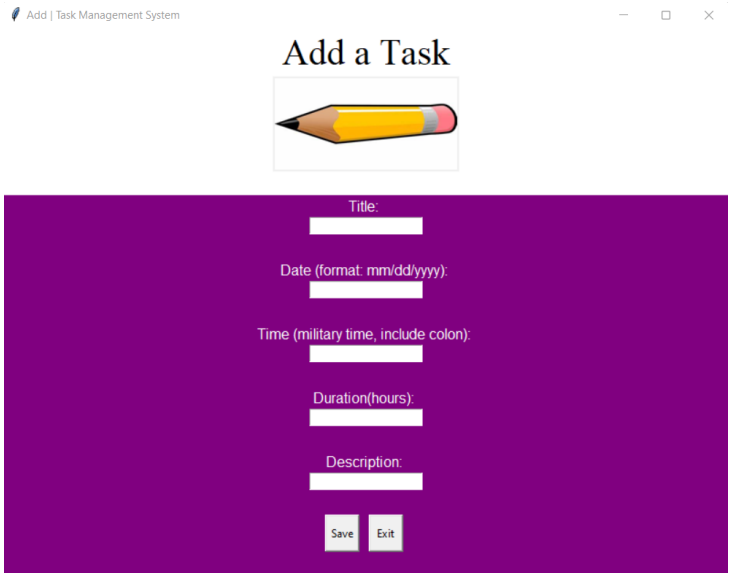


Figure 15 & 16: Add Page Layout Designs

Graphical User Interface Design

Add Page

Python Code for the Design Implementation:

```
'''
Team Purple
Task Management System
Add Page
'''

import tkinter as tk
from tkinter import *
from PIL import Image, ImageTk
import csv
import Class as TC
from tkinter import messagebox

def AddFunc(window = None):
    # Creating the adujstable window
    '''window = tk.Tk()
    window.title('Add | Task Management System')
    window.geometry('800x500')
    window.grid_rowconfigure(1, weight=1)
    window.grid_columnconfigure(0, weight=1)'''
    if window is None:
        window = tk.Tk()

    window.title('Add | Task Management System')
    window.geometry('800x600')
    window.grid_rowconfigure(1, weight=1)
    window.grid_columnconfigure(0, weight=1)
    window.configure(bg='purple')


    # Frames and Title
    Top = tk.Frame(window, bg = 'white', height = 100, width = 1000)
    Top.pack (fill=BOTH, expand=True)
    frame1 = tk.Frame(window, bg = 'purple') ##title
    frame2 = tk.Frame(window, bg = 'purple')##date
    frame3 = tk.Frame(window, bg = 'purple')##time
    frame4 = tk.Frame(window, bg = 'purple')##duration
    frame5 = tk.Frame(window, bg = 'purple')##Description
    frame6 = tk.Frame(window, bg = 'purple')##buttons
    frame1.pack (fill=BOTH, expand=True)
    frame2.pack (fill=BOTH, expand=True)
    frame3.pack (fill=BOTH, expand=True)
    frame4.pack (fill=BOTH, expand=True)
    frame5.pack (fill=BOTH, expand=True)
    frame6.pack(fill=BOTH, expand=True, padx=345)

    pageTitle = tk.Label(Top, text = 'Add a Task', fg = 'Black', bg = 'white', font = ('Times New Roman', 30))
    pageTitle.pack()

    # Image
    img = Image.open("AddImage.jpg") # load image
    resized_image = img.resize((200,100)) # resize, remove structural padding # Image.Resampling.LANCZOS
```

```
    # Image
    img = Image.open("AddImage.jpg") # load image
    resized_image = img.resize((200,100)) # resize, remove structural padding # Image.Resampling.LANCZOS
    new_image = ImageTk.PhotoImage(resized_image)# convert to photoimage
    imageLabel = Label(Top, image = new_image) #display the image
    imageLabel.pack()


    # Labels
    title = tk.Label(frame1, text = 'Title: ', fg = 'white', bg = 'purple', font = ('Bold', 12))
    date = tk.Label(frame2, text = 'Date (format: mm/dd/yyyy): ', fg = 'white', bg = 'purple', font = ('Bold', 12))
    time = tk.Label(frame3, text = 'Time (military time, include colon): ', fg = 'white', bg = 'purple', font = ('Bold', 12))
    duration = tk.Label(frame4, text = 'Duration(hours): ', fg = 'white', bg = 'purple', font = ('Bold', 12))
    description = tk.Label(frame5, text = 'Description: ', fg = 'white', bg = 'purple', font = ('Bold', 12))

    # Entries
    titleEnt = tk.Entry(frame1)
    dateEnt = tk.Entry(frame2)
    timeEnt = tk.Entry(frame3)
    durationEnt = tk.Entry(frame4)
    descriptionEnt = tk.Entry(frame5)

    title.pack()
    titleEnt.pack()
    date.pack()
    dateEnt.pack()
    time.pack()
    timeEnt.pack()
    duration.pack()
    durationEnt.pack()
    description.pack()
    descriptionEnt.pack()


    def Save():
        ##get the current user from text file
        with open ('CurrentUser.txt', 'r') as f:
            U = f.read()
        f.close()

        titleInp = titleEnt.get()
        dateInp = dateEnt.get()
        timeInp = timeEnt.get()
        durationInp = durationEnt.get()
        descrInp = descriptionEnt.get()

        try:
            if dateInp.find("/"):
                d = dateInp.split("/")
```

```
    def Save():
        ##get the current user from text file
        with open ('CurrentUser.txt', 'r') as f:
            U = f.read()
        f.close()

        titleInp = titleEnt.get()
        dateInp = dateEnt.get()
        timeInp = timeEnt.get()
        durationInp = durationEnt.get()
        descrInp = descriptionEnt.get()

        try:
            if dateInp.find("/"):
                d = dateInp.split("/")
                t=timeInp.split(":")
                condition1 = (int(d[-1]) >= 2022 and int(d[-1]) < 2033)
                condition2 = (len(timeInp)==5 or len(timeInp)==4)
                if condition1 and condition2:
                    infoList = [U, titleInp, dateInp, timeInp, durationInp, descrInp]
                    with open ('taskList.csv', 'a', newline = '') as file:
                        csvWriter = csv.writer(file)
                        csvWriter.writerow(infoList)
                    window.destroy()
                elif (int(d[-1]) < 2022 or int(d[-1]) > 2033):
                    messagebox.showwarning('Invalid Input', 'Please enter a valid date between year 2022 and 2032.')

                else:
                    messagebox.showwarning('Invalid Input', 'Please enter a valid date and time following the given formats.')
        except:
            messagebox.showwarning('Something went wrong.', 'Please ensure your input follows the formats provided.')


    def Exit():
        window.destroy()


    # Exit Button
    save = tk.Button(frame6, text = 'Save', width = 4, height = 2, command = Save).grid(column=0, row=0, padx=10)
    exitBut = tk.Button(frame6, text = 'Exit', width = 4, height = 2, command = Exit).grid(column=1, row=0)

    window.mainloop()


if __name__ == "__main__":
    AddFunc()
```

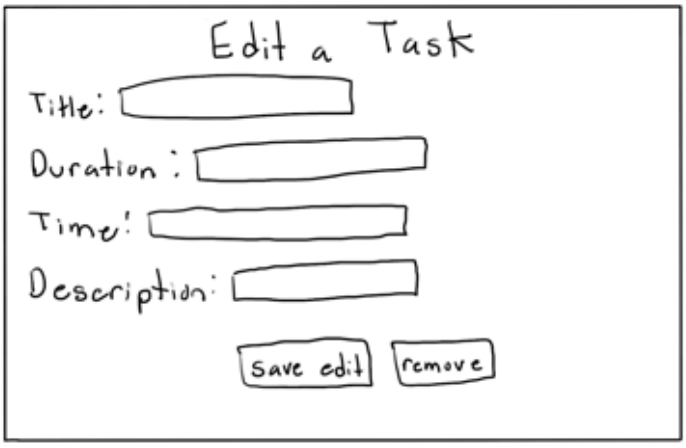
Figure 17: Add Page Code Implementation

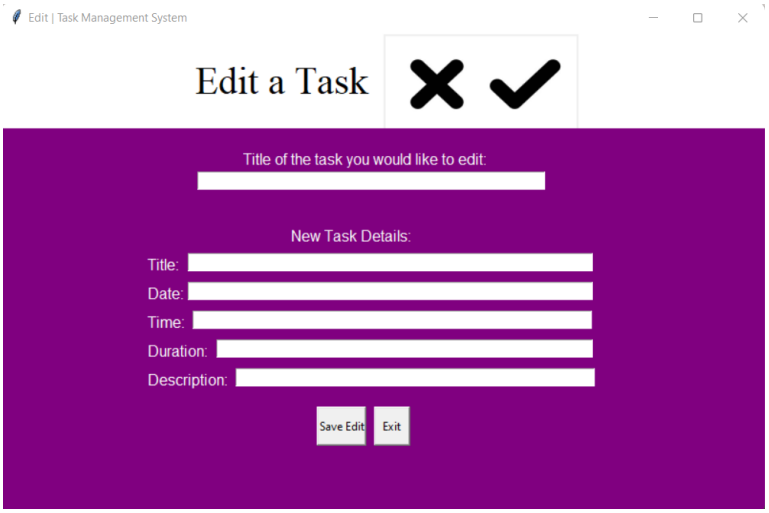
Graphical User Interface Design

Edit Page

The edit page design idea is very similar to the add page in order to provide a cohesive interface design. It includes a descriptive title, “Edit a Task”, at the top indicating the page. There will be entry boxes labeled according to their purpose of title, duration, time, or description. In our design, the entries will be followed by save and exit buttons. The ultimate implementation includes an additional entry field of data as well as an image in the banner at the top. The background is purple.

Design Idea:





Implementation of Design:

Figure 18 & 19: Edit Page Layout Designs

Graphical User Interface Design

Edit Page

Python Code for the Design Implementation:

```
'''
Team Purple
Task Management System
Edit Page
'''

import tkinter as tk
from tkinter import messagebox
from tkinter import *
from PIL import Image, ImageTk
import csv
import Class

##TMS Edit Page allows a user to edit a task's details
##invluding title, time, date, duration, and description.


def EditFunc(window = None):


    # Creating the adujstable window
    '''window = tk.Tk()
    window.title('Edit | Task Management System')
    window.geometry('800x500')
    window.grid_rowconfigure(1, weight=1)
    window.grid_columnconfigure(0, weight=1)'''

    if window is None:
        window = tk.Tk()

    window.title('Edit | Task Management System')
    window.geometry('800x500')
    window.grid_rowconfigure(1, weight=1)
    window.grid_columnconfigure(0, weight=1)

    # Frames and Title
    Top = tk.Frame(window, bg = 'white', height = 100, width = 1000)
    Top.grid (row = 0, sticky = 'nsew')
    Body = tk.Frame(window, bg = 'purple', height = 400, width = 1000)
    Body.grid(row = 1, sticky = 'nsew')

    title = tk.Label(Top, text = 'Edit a Task', fg = 'Black', bg = 'white', font = ('Times New Roman', 30))
    title.place(x = 200, y = 25)

    # Image
    img = Image.open("EditPage.png") # load image
    resized_image = img.resize((200,100)) # resize, remove structural padding #  Image.Resampling.LANCZOS
    new_image = ImageTk.PhotoImage(resized_image)# convert to photoimage
    label = Label(Top, image = new_image) #disply the image
    label.place(x = 400, y = 2)

    # Labels
```

```
    # Labels
    title = tk.Label(Body, text = 'Title of the task you would like to edit: ', fg = 'white', bg = 'purple', font = ('Bold', 12))
    title.place(x = 250, y = 20)
    details = tk.Label(Body, text = 'New Task Details: ', fg = 'white', bg = 'purple', font = ('Bold', 12))
    details.place(x = 300, y = 100)
    newTitle = tk.Label(Body, text = 'Title: ', fg='white', bg='purple', font=('Bold', 12))
    newTitle.place(x = 150, y = 130)
    date = tk.Label(Body, text = 'Date: ', fg = 'white', bg = 'purple', font = ('Bold', 12))
    date.place(x = 150, y = 160)
    time = tk.Label(Body, text = 'Time: ', fg = 'white', bg = 'purple', font = ('Bold', 12))
    time.place(x = 150, y = 190)
    duration = tk.Label(Body, text = 'Duration: ', fg = 'white', bg = 'purple', font = ('Bold', 12))
    duration.place(x = 150, y = 220)
    description = tk.Label(Body, text = 'Description: ', fg = 'white', bg = 'purple', font = ('Bold', 12))
    description.place(x = 150, y = 250)

    # Entries
    titleEnt = tk.Entry(Body, width = 60)
    titleEnt.place(x = 205, y = 45)
    newTitleEnt =tk.Entry(Body, width = 70)
    newTitleEnt.place(x = 195, y = 130)
    dateEnt = tk.Entry(Body, width = 70)
    dateEnt.place(x = 195, y = 160)
    timeEnt = tk.Entry(Body, width = 69)
    timeEnt.place(x = 200, y = 190)
    durationEnt = tk.Entry(Body, width = 65)
    durationEnt.place(x = 225, y = 220)
    descriptionEnt = tk.Entry(Body, width = 62)
    descriptionEnt.place(x = 245, y = 250)

    ##get the current user from text file
    with open ('CurrentUser.txt', 'r') as f:
        username = f.read()
    f.close()

    # Function
    def ExitWin():
        window.destroy()
    def EditFunc():
        L = []
        found=False
        TaskToEdit = titleEnt.get()
        titleNew = newTitleEnt.get()
        dateNew = dateEnt.get()
        timeNew = timeEnt.get()
        durationNew = durationEnt.get()
        descrNew = descriptionEnt.get()

        #use elif when checking for valid title

        with open ('taskList.csv', 'r') as file:
            csvReader = csv.reader(file)
```

```
    # Function
    def ExitWin():
        window.destroy()
    def EditFunc():
        L = []
        found=False
        TaskToEdit = titleEnt.get()
        titleNew = newTitleEnt.get()
        dateNew = dateEnt.get()
        timeNew = timeEnt.get()
        durationNew = durationEnt.get()
        descrNew = descriptionEnt.get()

        #use elif when checking for valid title

        with open ('taskList.csv', 'r') as file:
            csvReader = csv.reader(file)
            for row in csvReader:
                if (row[0] == username and row[1] == TaskToEdit): #finding old task info to edit, leave out of list
                    found=True
                    L.append([username, titleNew, dateNew, timeNew, durationNew, descrNew]) #add new task info to list
                else:
                    L.append(row)

        file.close()

        if not found:
            messagebox.showwarning('Edit Unsuccessful', f'Task {TaskToEdit} not found.')
        if found:
            with open ('taskList.csv', 'w', newline = '') as file: #write all task info in List back in to taskList.csv
                csvWriter = csv.writer(file)
                csvWriter.writerows(L)
            window.destroy()
            messagebox.showwarning('Edit Successful', f'Task {TaskToEdit} details edited.')

    # Button
    save = tk.Button(Body, text = 'Save Edit', width = 6, height = 2, command=EditFunc).place(x = 330, y = 290)
    exitBut = tk.Button(Body, text = 'Exit', width = 4, height = 2, command=ExitWin).place(x = 390, y = 290)


    window.mainloop()

if __name__ == "__main__":
    EditFunc()
```

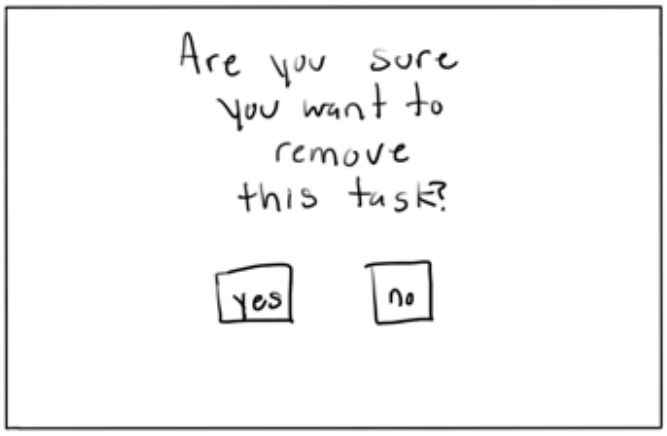
Figure 20: Main Page Code Implementation

Graphical User Interface Design

Remove Page

The remove page design is for it to be a simple interface/pop-up. It will contain the text “Are you sure you want to remove this task?” and two buttons, yes or no. Like the other interfaces, our implementation includes a title at the top. The implementation also has a box in the center for the task selected and the yes/no buttons are color coded so it is harder for the user to misclick.

Design Idea:



Implementation of Design:

Figures 21 & 22: Remove Page Layout Designs

Graphical User Interface Design

Remove Page

Python Code for the Design Implementation:

```
'''
Team Purple
Task Managemnet System
Remove Page GUI
'''
import tkinter as tk
from tkinter import messagebox
import csv
import Class

def RemoveWin(window = None):

    if window is None:
        window = tk.Tk()

    # Creating the window
    #window = tk.Tk()
    window.title('Remove a Task')
    window.geometry('500x200')
    window.resizable(width = False, height = False)

    # Frames
    frame = tk.Frame(window, bg = 'lightgray', height = 300, width = 500)
    frame.place(x = 1, y = 1)
    task = tk.Entry(window, width = 50)
    task.place(x = 85, y = 72)

    # Text
    header = tk.Label(frame, text = 'Remove a Task', fg = 'black', bg = 'lightgray', font = ('Times New Roman', 20))
    header.place(x = 155, y = 5)
    text = tk.Label(frame, text = 'Please enter the title of the task you would like to remove: ', fg = 'black', bg = 'lightgray', font = ('Times New Roman', 10))
    text.place(x = 75, y = 50)


    # Gets username
    with open ('CurrentUser.txt', 'r') as f:
        username = f.read()
    f.close()

    # Functions

    def RemoveFunc():
        found=False
        L = []
        TitleToRemove = task.get()
        with open ('taskList.csv', 'r') as file:
            csvReader = csv.reader(file)
            for row in csvReader:
                #print (row[0], row[1], username, TitleToRemove)
                if (row[0] == username and row[1] == TitleToRemove):
                    found=True
                else:
                    L.append(row)
        file.close()

        if not found:
            messagebox.showwarning('Remove Unsuccessful', f'Task {TitleToRemove} not found.')
        if found:
            with open ('taskList.csv', 'w', newline = '') as file:
                csvWriter = csv.writer(file)
                csvWriter.writerows(L)
                window.destroy()
            messagebox.showwarning('Remove Successful', f'Task {TitleToRemove} removed.')

    def CancelFunc():
        window.destroy()

    # Buttons
    yes = tk.Button(frame, text = 'Remove', bg = 'green',fg = 'black', command=RemoveFunc)
    yes.place(x =  170, y = 120)
    no = tk.Button(frame, text = 'Cancel', bg = 'red', fg = 'black', command = CancelFunc )
    no.place(x = 240, y = 120)

    window.mainloop()

if __name__ == "__main__":
    RemoveWin()
```

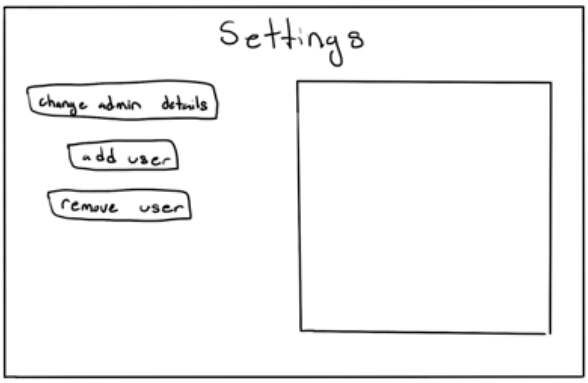
Figure 23: Remove Page Code Implementation

Graphical User Interface Design

Settings Page

The settings page interface design has an identifying title of Settings at the top and three buttons of change admin details, add user, and remove user. These interface design ideas are still the same in the implementation. The implementation also includes username and password entry boxes as well as an image. Pressing the buttons will display message boxes that give useful information on how to interact with the interface. If nothing is entered into the boxes, the message appears saying to input information and then click the button again to confirm changes. Pressing the "return to main Page" button closes the window and links back to the main page.

Design Idea:



Implementation of Design:

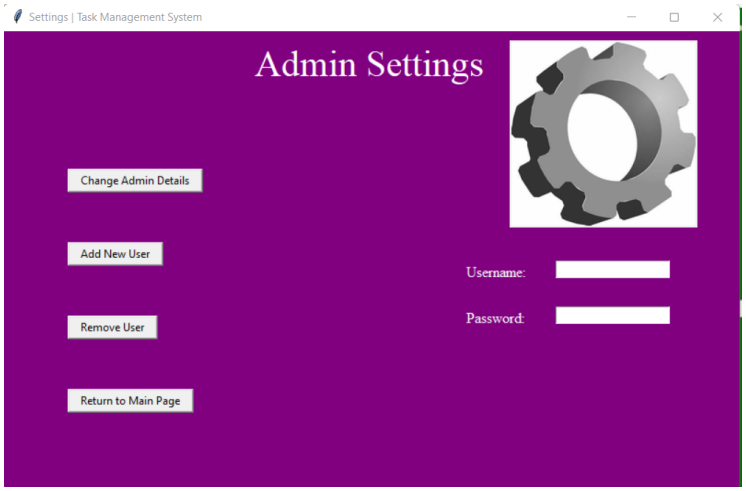


Figure 24 & 25: Settings Page Layout Design

Graphical User Interface Design

Settings Page

Python Code for the Design Implementation:

```
import tkinter as tk
from tkinter import *
from PIL import ImageTk, Image
import csv
import Class
from tkinter import messagebox


##TMS Administrator Settings page
##Allows the admin user to change user details such as username and password

AdminChanges = False ##flag determines whether changes have been made

def returnToMain(AdminChanges, window):
    if AdminChanges == True:
        messagebox.showwarning("Changes have been made to this account.","Changes have been made to this account. "+
                               "You may have to log out and log back in again for changes to appear.")
    window.destroy()


def addUser(usernEnt, passEnt):
    ##adds a new user
    user=usernEnt.get()
    password=passEnt.get()
    adminUserNew=usernEnt.get()
    adminPassNew=passEnt.get()

    if user == '' and password == '':
        messagebox.showinfo('Add New User', "In order to add a new user, enter details in the boxes provided."
                            +" Then, press this button again to enter changes.")
    elif len(adminUserNew)<8 or len(adminPassNew)<8:
        messagebox.showwarning('Invalid Input', 'The username and password must be at least 8 characters long.')

    else:
        if messagebox.askyesno('Add New User', 'Add this user?'):
            with open ('CurrentUser.csv','r') as file:
                csvReader=csv.reader(file)
                for SingleRow in csvReader:
                    CurrentAdmin = SingleRow[0]
            file.close()

            with open('userLoginInfo.csv','a',newline='') as file:
                csvWriter=csv.writer(file)
                csvWriter.writerow([user,password,"FALSE"])
            file.close
            messagebox.showinfo("Action Successful", "New user has been added successfully.")


def changeAdmin(usernEnt, passEnt):
    ##changes the admin username and password and replaces all instances of username in files
    adminUserNew=usernEnt.get()
```

```
            file.close
            messagebox.showinfo("Action Successful", "New user has been added successfully.")


def changeAdmin(usernEnt, passEnt):
    ##changes the admin username and password and replaces all instances of username in files
    adminUserNew=usernEnt.get()
    adminPassNew=passEnt.get()

    if adminUserNew == '' and adminPassNew == '':
        messagebox.showinfo('Change Username and Password', "In order to change your username and passowrd, enter details in the boxes provided."
                            +" Then, press this button again to enter changes.")
    elif len(adminUserNew)<8 or len(adminPassNew)<8:
        messagebox.showwarning('Invalid Input', 'The username and password must be at least 8 characters long.')

    else:
        if messagebox.askyesno('Change Username and Passowrd', 'Change Admin username and password?'):
            L=[] ## 2D List to hold contents of file
            U = ''
            ##get the current user from text file
            with open('CurrentUser.txt', 'r') as f:
                U = f.read()
            f.close()

            ##move contents of file to list unless line contains current admin (user)
            with open('userLoginInfo.csv','r') as file:
                csvReader=csv.reader(file)
                for member in csvReader:
                    if(member[-1]=='TRUE'): ##search for Admin
                        continue ##if admin, do not append line to list
                    else:
                        L.append(member) ##append line of file to list
                L.append([adminUserNew,adminPassNew,'TRUE']) ##add new admin info at the end of list
            file.close()

            with open('userLoginInfo.csv','w',newline='') as file:
                csvWriter=csv.writer(file)
                csvWriter.writerows(L) ##rewrite file with updated info
            file.close()

            ##rewrite tasklist with updated admin username
            Contents=[]
            with open('taskList.csv', 'r') as fi:
                csvReader = csv.reader(fi)
                for task in csvReader:
                    if (task[0]==U):
                        task[0] = adminUserNew
                        Contents.append(task)
                    else:
                        Contents.append(task)
            fi.close()
```

```
            fi.close()

            with open('taskList.csv', 'w', newline='') as file:
                csvWriter = csv.writer(file)
                csvWriter.writerows(Contents)
            file.close()

            ##update current user file
            with open('CurrentUser.txt','w') as f:
                f.write(adminUserNew)
            f.close()
            messagebox.showinfo('Successful Change', 'The Admin username and password has been changed.'+
                                "You will have to log out and log back in again to apply changes to main page.")

def removeUser(usernEnt, passEnt):
    user=usernEnt.get()
    password=passEnt.get()

    U = ''
    ##get the current user from text file
    with open('CurrentUser.txt', 'r') as f:
        U = f.read()
    f.close()

    if user=="" and password =="":
        messagebox.showinfo('User not found.', 'Please enter the username and password of an existing user to delete their account.'+
                            " Then, press this button to finalize change.")
    else:
        if messagebox.askyesno('Remove User', 'Remove this user and their tasks for good?'):
            Contents=[]

            ##move contents of file to list unless line contains user to be removed
            with open('userLoginInfo.csv','r') as file:
                csvReader=csv.reader(file)
                for member in csvReader:
                    if(member[0]==user and member[1]==password):
                        continue
                    else:
                        Contents.append(member)
            file.close()

            with open('userLoginInfo.csv','w',newline='') as file: ##write over file excluding removed user
                csvWriter=csv.writer(file)
                csvWriter.writerows(Contents)
            file.close()

            ##rewrite tasklist excluding the deleted user's tasks
            L=[]
            with open('taskList.csv', 'r') as file:
                csvReader = csv.reader(file)
                for task in csvReader:
                    if (task[0]==user):
```

```
            ##rewrite tasklist excluding the deleted user's tasks
            L=[]
            with open('taskList.csv', 'r') as file:
                csvReader = csv.reader(file)
                for task in csvReader:
                    if (task[0]==user):
                        continue
                    else:
                        L.append(task)
            file.close()

            with open('taskList.csv', 'w', newline='') as file:
                csvWriter = csv.writer(file)
                csvWriter.writerows(L)
            file.close()

            messagebox.showinfo('Action Successful', 'The user has been successfully removed.')

def Set(window = None):

    '''window=tk.Tk()
    window.title('Settings')
    window.geometry('800x500')
    window.configure(bg = "Purple")'''

    if window is None:
        window = tk.Tk()

    window.title('Settings | Task Management System')
    window.geometry('800x500')
    window.grid_rowconfigure(1, weight=1)
    window.grid_columnconfigure(0, weight=1)

    BodyFrame=tk.Frame(window,bg='purple',height=1000,width=1000)
    BodyFrame.pack()

    # Image
    img=Image.open("SetPage.jpg")
    resized_image=img.resize((200,200), Image.Resampling.LANCZOS)
    new_image=ImageTk.PhotoImage(resized_image)
    label=Label(BodyFrame, image=new_image)
    label.place(x=550,y=10)

    titleLab=tk.Label(BodyFrame,text="Admin Settings", bg='purple', fg = 'white', font = ('Times New Roman', 30))
    titleLab.place(x=270,y=10)

    usernLab=tk.Label(BodyFrame,text="Username: ",bg='purple', fg = 'white', font = ('Times New Roman', 12))
    usernLab.place(x=500,y=250)

    usernEnt=tk.Entry(BodyFrame)
    usernEnt.place(x=600,y=250)
```

```
    if window is None:
        window = tk.Tk()

    window.title('Settings | Task Management System')
    window.geometry('800x500')
    window.grid_rowconfigure(1, weight=1)
    window.grid_columnconfigure(0, weight=1)

    BodyFrame=tk.Frame(window,bg='purple',height=1000,width=1000)
    BodyFrame.pack()

    # Image
    img=Image.open("SetPage.jpg")
    resized_image=img.resize((200,200), Image.Resampling.LANCZOS)
    new_image=ImageTk.PhotoImage(resized_image)
    label=Label(BodyFrame, image=new_image)
    label.place(x=550,y=10)

    titleLab=tk.Label(BodyFrame,text="Admin Settings", bg='purple', fg = 'white', font = ('Times New Roman', 30))
    titleLab.place(x=270,y=10)

    usernLab=tk.Label(BodyFrame,text="Username: ",bg='purple', fg = 'white', font = ('Times New Roman', 12))
    usernLab.place(x=500,y=250)

    usernEnt=tk.Entry(BodyFrame)
    usernEnt.place(x=600,y=250)

    passLab=tk.Label(BodyFrame, text="Password: ",bg='purple', fg = 'white', font = ('Times New Roman', 12))
    passLab.place(x=500,y=300)

    passEnt=tk.Entry(BodyFrame)
    passEnt.place(x=600,y=300)

    chgAdminBut=tk.Button(BodyFrame, text='Change Admin Details', padx = 10, command=lambda:changeAdmin(usernEnt, passEnt))
    chgAdminBut.place(x=70,y=150)

    adduserBut=tk.Button(BodyFrame, text="Add New User", padx = 10, command=lambda:addUser(usernEnt, passEnt))
    adduserBut.place(x=70,y=230)

    chgpassBut=tk.Button(BodyFrame, text="Remove User", padx = 10, command=lambda:removeUser(usernEnt, passEnt))
    chgpassBut.place(x=70,y=310)

    returnBut = tk.Button(BodyFrame, text = "Return to Main Page", padx=10, command=lambda:returnToMain(AdminChanges, window))
    returnBut.place(x=70,y=390)

    window.mainloop()


if __name__ == "__main__":
    Set()
```

Figure 26: Settings Page Code Implementation

Page Connections

The first page brought up for the user is the login page. Connected to the login page after successful verification is the main page. These pages are connected by calling the function of the main page python file within the login page python file, as seen in figure 23. The main page contains the connections for all of the remaining pages of Add, Remove, Edit, and Settings. These pages are connected to the main page in the same way by giving each button a command that call the function of the other page, as seen also in figure 23, for example. This allows the user to logically choose which page they need at any given time.

Login/Main Page Connection:

```
for member in csvReader:
    if member[0]==username and member[1]==password:
        found = True
        user=member
        Exit(window)
        M.mainFunc(member)
```

Button/Action Page Connections:

```
addButton = Button(taskFrame, text="Add New Task",
                   command=lambda:Add.AddFunc(window=Toplevel()))
editButton = Button(taskFrame, text="Edit Selected Task",
                    command=lambda:Edit.EditFunc(window=Toplevel()))
removeButton = Button(taskFrame, text="Remove Selected Task",
                      command=lambda:Remove.RemoveWin(window=Toplevel()))
```

Figure 23: Page Connections Examples

Figure 27: Examples of Page Connections

Data Storage

There are two separate csv files to store all of the necessary information of the Task Management System. Both csv files retain their information even after the program is closed so a user's task data is not lost. The first csv, called userLoginInfo, stores the username, password, and admin status for each user. The second csv, taskList, stores task information such as title, date, time, duration, and description by username. We also used a text file, CurrentUser.txt, to store the username of the currently logged in user. We stored and read information from these files using the code from figure 24 below.

Writing into a CSV:

```
with open('userLoginInfo.csv','w',newline='') as file:
    csvWriter=csv.writer(file)
    csvWriter.writerows(Contents)

infoList = [U, titleInp, dateInp, timeInp, durationInp, descrInp]

with open ('taskList.csv', 'a', newline = '') as file:
    csvWriter = csv.writer(file)
    csvWriter.writerow(infoList)
```

Figure 28: Example of Writing to a CSV File

Reading From a CSV

```
TASKFILE = "taskList.csv"

def logOut(window, user):
    if messagebox.askyesno('Log Out', 'Are you sure you want to log out?'):
        window.destroy()
        Login.LogPage()


def printTasksToTextBox(user,tasksTextBox):
    tasksTextBox.delete(1.0,END)
    num=1
    with open(TASKFILE, 'r') as file:
        csvReader=csv.reader(file)
        for member in csvReader:
            if member[0] == user[0]:
                string = f'{num}.) '
                for element in member: ##prints task details excluding username
                    if element != member[0]:
                        string = string+element+" | "
                string=string+"\n****************************************"
                tasksTextBox.insert(END, string)
                num+=1
```

Figure 29: Example of Reading From a CSV File

Writing To and Reading From a Text File

```
##save the current user to a text file
with open ('CurrentUser.txt', 'w') as f:
    f.write(str(member[0]))
f.close()


def Save():
    ##get the current user from text file
    with open ('CurrentUser.txt', 'r') as f:
        U = f.read()
    f.close()
```

Figure 30: Example of Writing to and Reading From a Text File

References

GUI Programming in Python

https://docs.python.org/3/library/calendar.html#module-calendar

https://www.plus2net.com/python/tkinter-rowconfigure.php

https://pythonguides.com/python-tkinter-listbox/

https://realpython.com/python-gui-tkinter/

https://www.plus2net.com/python/tkinter-rowconfigure.php

https://www.tutorialspoint.com/python/tk_relief.htm

Python GUI examples (Tkinter Tutorial) - Like

Geekshttps://pythonguides.com/python-tkinter-radiobutton/

Why is Button parameter "command" executed when declared? - Stack Overflow

https://coderslegacy.com/python/python-gui/python-tkinter-combobox/

https://stephenallwright.com/python-month-number/

https://docs.python.org/3/library/datetime.html#

https://likegeeks.com/python-gui-examples-tkinter-tutorial/#Add_radio_buttons_widgets

https://coderslegacy.com/python/tkinter-close-window/

https://www.geeksforgeeks.org/python-tkinter-text-widget/

Is there a way to clear all widgets from a tkinter window in one go without referencing them all directly? - Stack Overflow

https://pynative.com/python-get-the-day-of-week/

https://www.geeksforgeeks.org/how-to-install-pil-on-windows/amp/