

# 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.

## Table of Contents

Table of Figures .....	4
Project Description.....	5
GitHub Repository Address .....	6
Graphical User Experience Design Flowchart.....	7
Graphical User Experience Design.....	8-14
a. Login Page	
b. Main Window	
c. Action Pages	
i. Add Page; Search Page; Edit Page; Remove Page; Settings Page	
Graphical User Interface Design.....	15-30
a. Login Page	
b. Main Window	
c. Action Pages	
i. Add Page; Search Page; Edit Page; Remove Page; Settings Page	
Page Connections.....	31
Data Storage.....	32
References.....	33

## Table of Figures

1. Figure 1: Complete Flowchart.....	7
1. Figure 2: Login Page Flowchart.....	8
2. Figure 3: Main Page Flowchart.....	9
3. Figure 4: Add Page Flowchart.....	10
4. Figure 5: Search Page Flowchart.....	11
5. Figure 6: Edit Page Flowchart.....	12
6. Figure 7: Remove Page Flowchart.....	13
7. Figure 8: Settings Page Flowchart.....	14
8. Figures 9 & 10: Login Page Layout Designs.....	15
9. Figures 11 & 12: Main Page Layout Designs.....	16
10. Figures 13 & 14: Add Page Layout Designs.....	22
11. Figures 15 & 16: Search Page Layout Designs.....	23
12. Figures 17 & 18: Edit Page Layout Designs.....	25
13. Figures 19 & 20: Remove Page Layout Designs.....	27
14. Figures 21 & 22: Settings Page Layout Designs.....	29
15. Figure 23: Page Connections Examples.....	31
16. Figures 24: Data Storage Examples.....	32

## 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.

CMPT 120L Project\_Phase\_04\_TeamPurple

## GitHub Repository Address

[https://github.com/Abel-Scholl/CMPT102L\\_TaskManagementSystem\\_TeamPurple#cmpt102l\\_taskmanagementsystem\\_teampurple](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 authorized, 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 calendar 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.

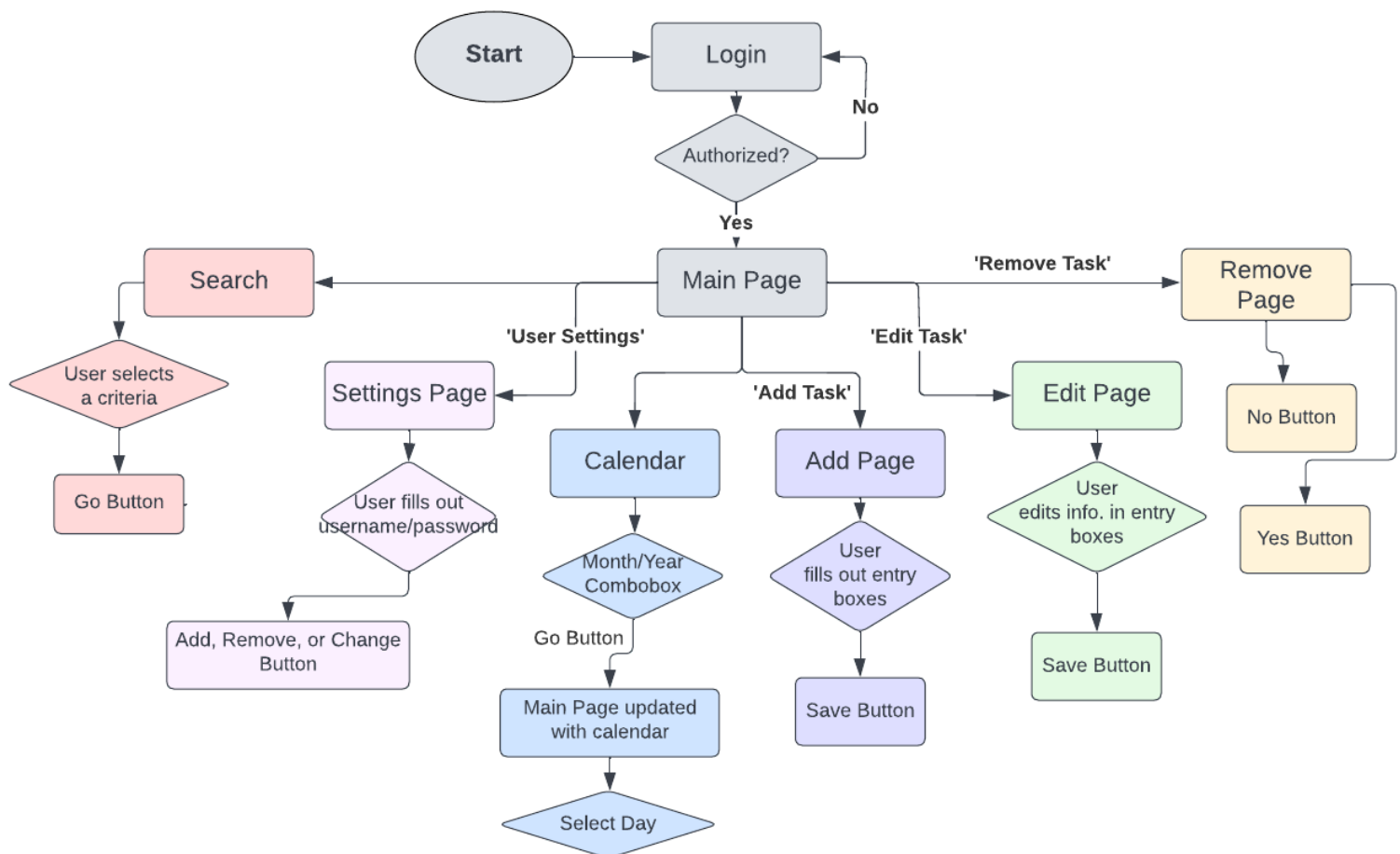


Figure 1: Complete Flowchart (Lucidchart)

## 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.

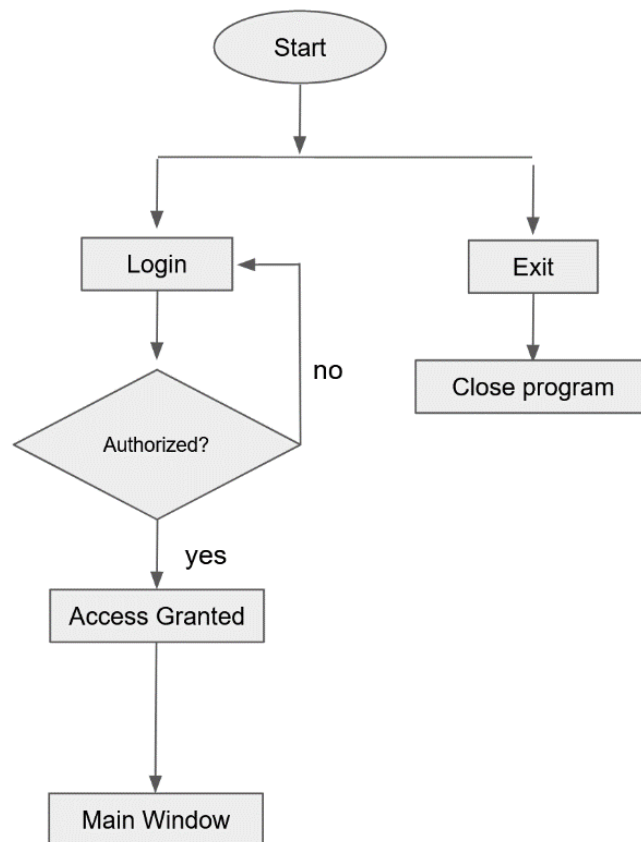


Figure 2: Login Page Flowchart (Lucidchart)



# 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.

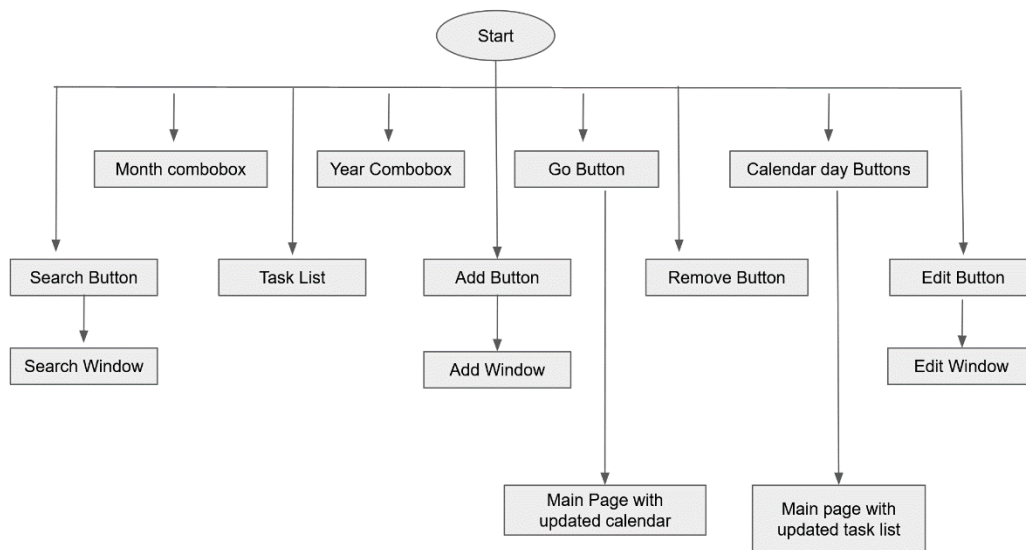


Figure 3: Main Page Flowchart (Lucidchart)

## Graphical User Experience Design

### *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).

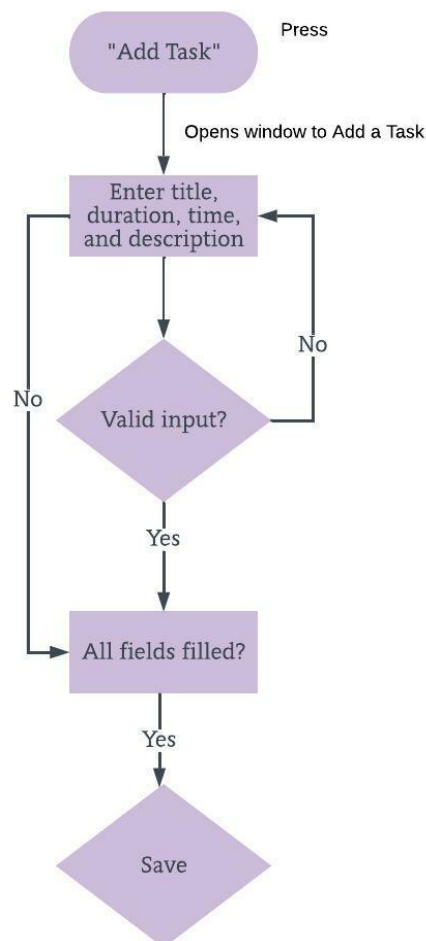


Figure 4: Add Page Flowchart (Lucidchart)

## Graphical User Experience Design

### *Search Page*

The search page allows users to search for a task(s) by using the categories of title, time, 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 page will output a list of any tasks that match the inputted information.

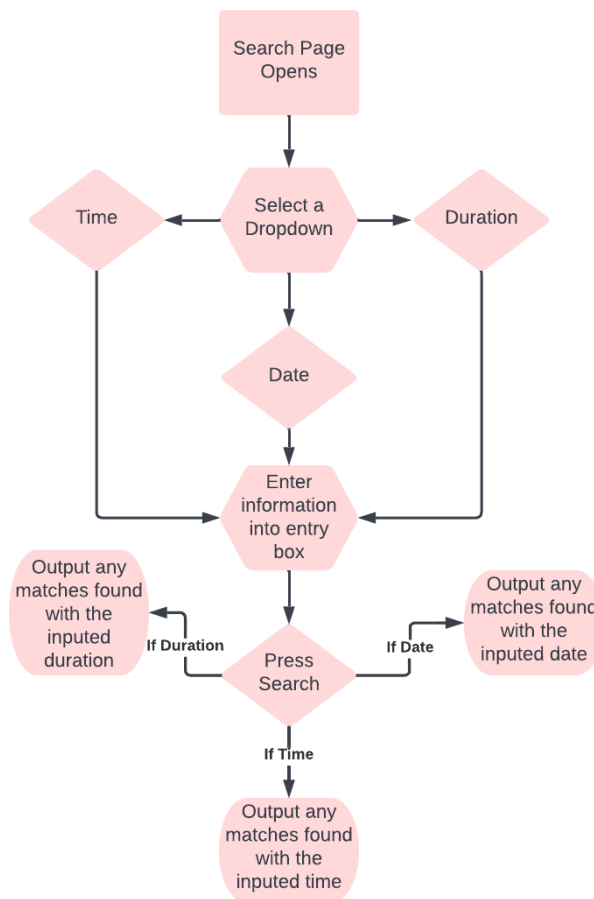


Figure 5: Search Page Flowchart (Lucidchart)

## 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).

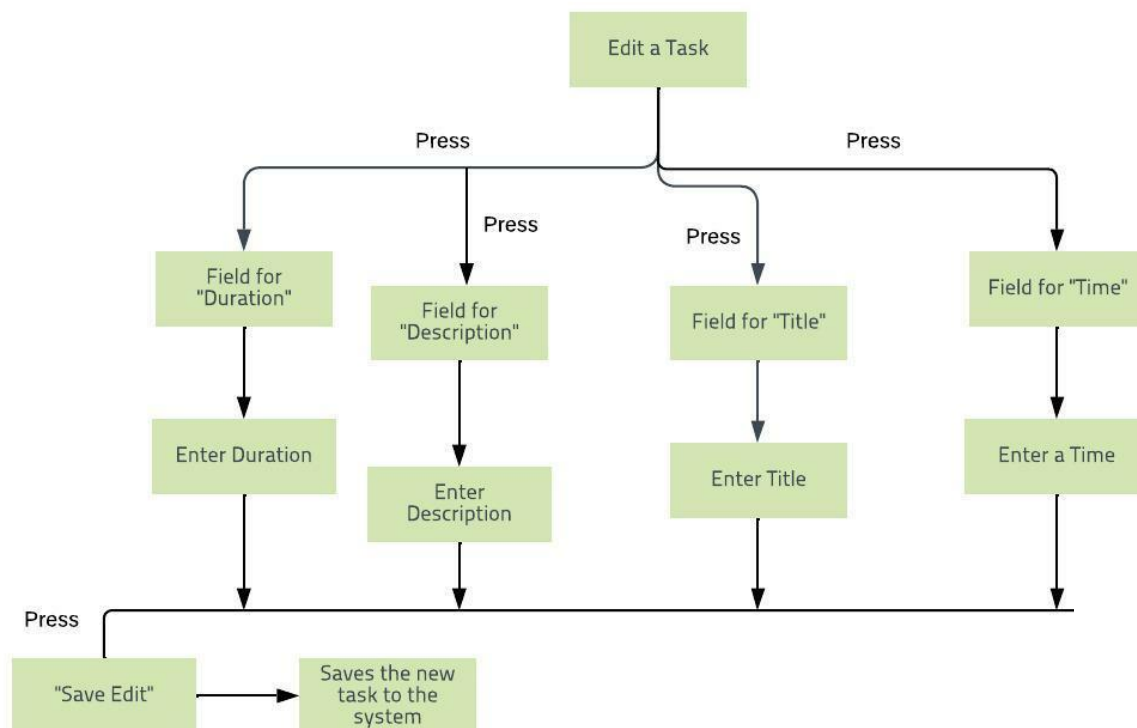


Figure 6: Edit Page Flowchart (Lucidchart)

## 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.

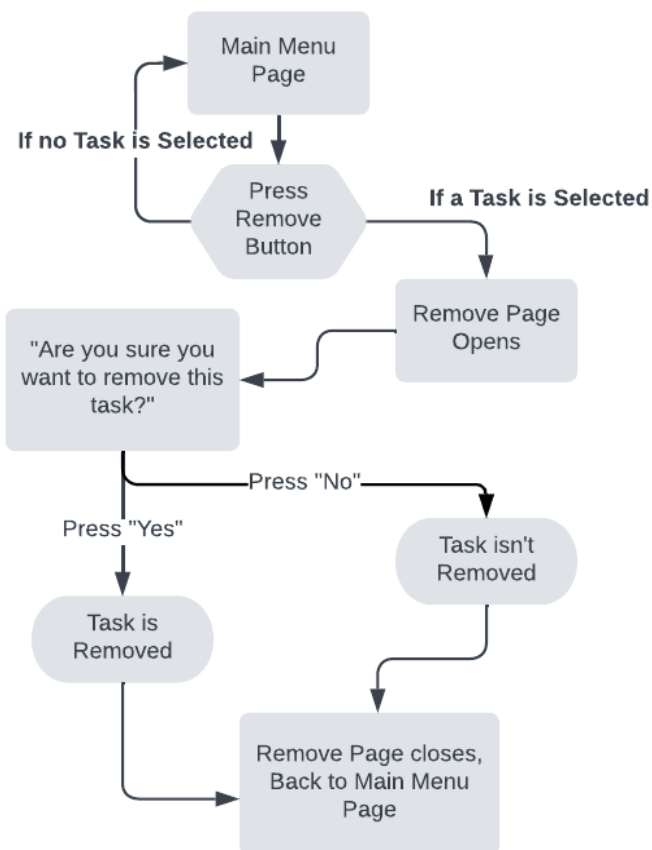


Figure 7: Remove Page Flowchart (Lucidchart)

## 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).

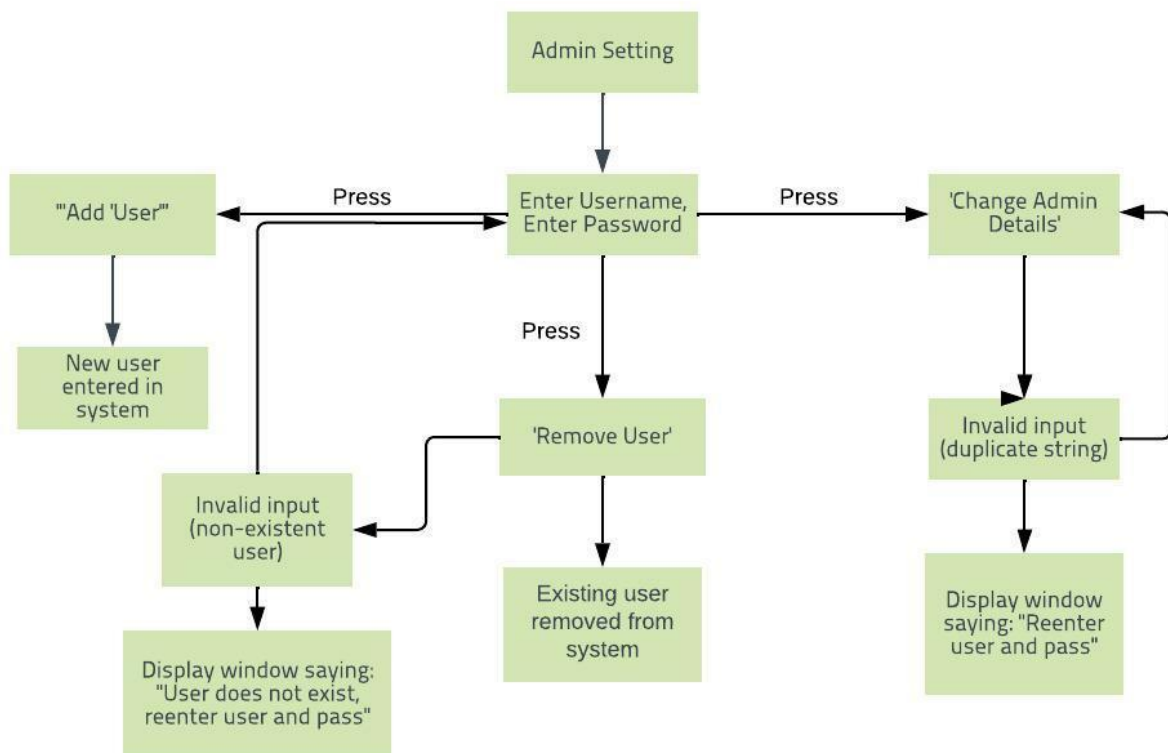


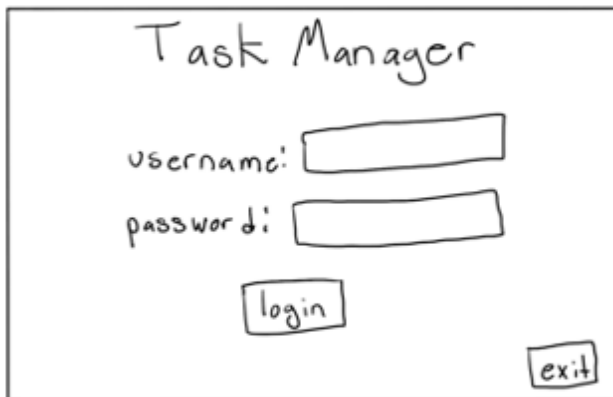
Figure 8: Settings Page Flowchart (Lucidchart)

## Graphical User Interface Design

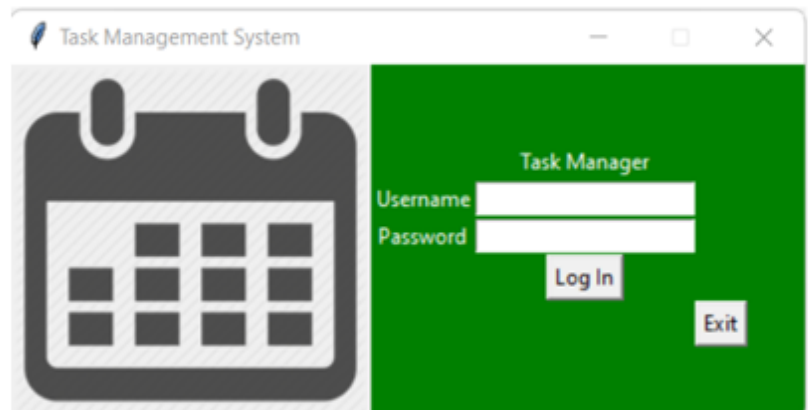
### *Login Page*

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:

```
import tkinter as tk

BG_COLOR = "Green"
FG_COLOR = "White"

window=tk.Tk()
window.title('Task Management System')
window.geometry('220x130')

BodyFrame=tk.Frame(window, bg=BG_COLOR, height=200, width=130)
BodyFrame.grid(row=1, sticky="nsew")

window.grid_rowconfigure(1, weight=1) ##allows user to adjust window

window.grid_columnconfigure(0, weight=1)

titleLabel = tk.Label(BodyFrame, text="Task Manager", bg = BG_COLOR, fg= FG_COLOR)
titleLabel.grid(column=2,row=0)

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

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

##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=login)
loginButton.grid(column=2,row=5)

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

window.mainloop()
```

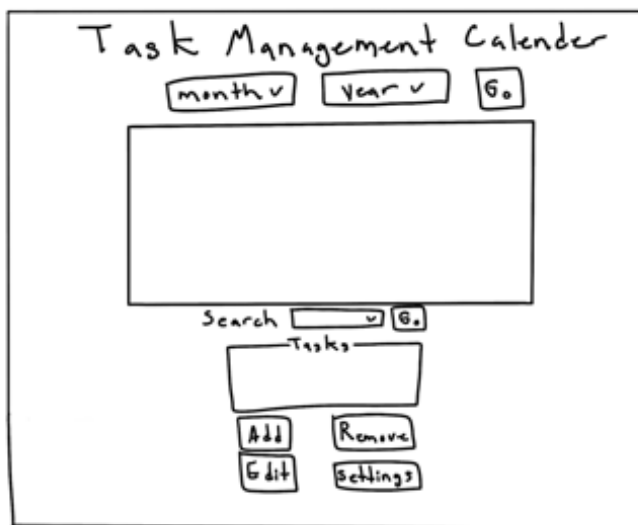


## Graphical User Interface Design

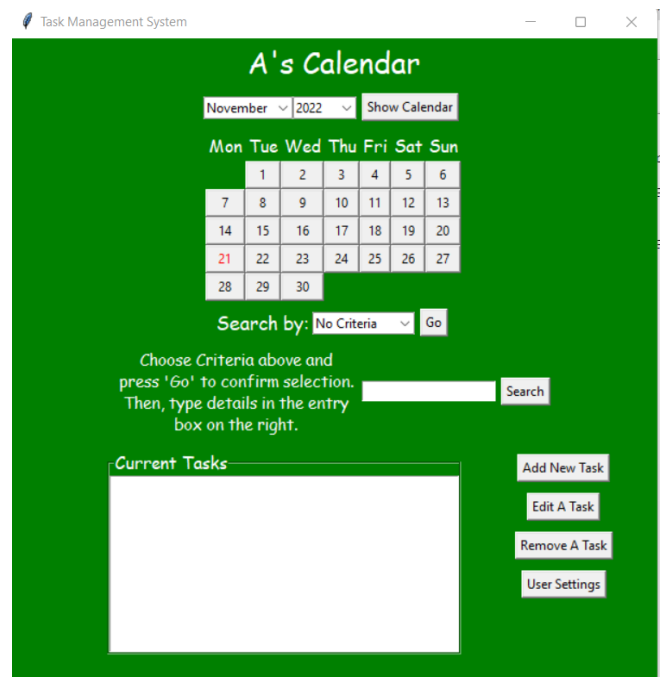
### *Main Window*

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 11 & 12: Login Page Layout Designs

# Graphical User Interface Design

## *Main Window*

### Python Code for the Desgin Implementation:

```
import datetime
from tkinter import *
from tkinter.ttk import *

window = Tk()

window.title('Task Management System')
window.geometry('500x500')

BG_COLOR = "Green"
FG_COLOR = "White"

##Months
monthsFrame = Frame(master=window)#, bg="Green")
selected = StringVar()

monthsCombo = Combobox(monthsFrame)
monthsCombo['values']= ("January", "February", "March", "April",
                        "May", "June", "July", "August",
                        "September", "October", "November", "December")
monthsCombo.current(1)
monthsCombo.pack(side=TOP)

monthsFrame.grid(column=0, row=0)

##Calendar
calendarFrame=Frame(master=window)#, bg="Green")
dayNum=0
days = ["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"]
dateToday = datetime.date.today()

weekdayLabel = Label()
for i in range(5):
    for j in range(7):
        ##frame = tk.Frame(master=window)
        if dayNum > 30:
            break
        else:
            dayNum+=1
            button = Button(master=calendarFrame, text=f"{dayNum}")
            button.grid(row=i, column=j, sticky='nsew')

calendarFrame.grid(column=0, row=1)
```

```

import datetime
from tkinter import *
from tkinter.ttk import *

##CURRENT_DAY = datetime.date.today()

window = Tk()

window.title('Task Management System')
window.geometry('550x500')
##window.configure(bg="Green")

BG_COLOR = "Green"
FG_COLOR = "White"

##Welcome Banner
welcFrame = Frame(master=window)
welcomeBanner = Label(welcFrame, text="[User]'s Calendar",
                      font = ("Comic Sans MS",20))
welcomeBanner.pack(side=LEFT)
welcFrame.pack()

##Months and Years as combo boxes + go button
monthsFrame = Frame(master=window)#, bg="Green")

monthsCombo = Combobox(monthsFrame)
monthsCombo['values']= ("January", "February", "March", "April",
                        "May", "June", "July", "August",
                        "September", "October", "November", "December")
monthsCombo.current(1)
monthsCombo.pack(side=LEFT)

yearsCombo = Combobox(monthsFrame)
yearsCombo['values']= ("2022", "2023", "2024", "2025",
                      "2026", "2027", "2028", "2029",
                      "2030", "2031", "2032", "2033")
yearsCombo.current(1)
yearsCombo.pack(side=LEFT)

##changes calendar to reflect selected info
goButtonCal = Button(monthsFrame, text="Go")
goButtonCal.pack(side=LEFT)

monthsFrame.pack()

```

```

calendarFrame=Frame(master=window)#, bg="Green")
dayNum=0
days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]
dateToday = datetime.date.today()

##creates a label for each weekday starting with monday
for d in range(len(days)):
    weekdayLabel = Label(calendarFrame, text=days[d], font=("Comic Sans MS", 12))
    weekdayLabel.grid(column=d, row=0)

    m=True##placeholder for datetime condition
for i in range(5):
    for j in range(7):
        ##frame = tk.Frame(master=window)
        if dayNum > 30:
            break
        else:
            if m==True:
                label= Label(calendarFrame, text=" ")
                label.grid(row=i+1, column=j, sticky='nsew')
                m=False
            else:
                dayNum+=1
                button = Button(calendarFrame, text=f"{dayNum}")
                button.grid(row=i+1, column=j, sticky='nsew')

calendarFrame.pack()

##Frame for search function
searchFrame = Frame(master=window)
searchLabel=Label(searchFrame, text="Search by:", font=("Comic Sans MS", 10))
searchLabel.pack(side=LEFT)

searchByCombo = Combobox(searchFrame)
searchByCombo['values']= ("Month", "Day", "Year", "Time",
                          "Title", "Duration", "Description")
searchByCombo.current(1)
searchByCombo.pack(side=LEFT)

goButtonSearch = Button(searchFrame, text="Go")
goButtonSearch.pack(side=LEFT)

searchFrame.pack()

taskFrame = Frame(master=window)
taskLabelFrame = LabelFrame(taskFrame, text='Current Tasks')

##if tasks exist for that search. else, label says: "There are no tasks scheduled for this day"
task = Radiobutton(taskLabelFrame, text="placeholder") ##find way to click away and clear selection?
task.pack()
taskLabelFrame.pack()

addButton = Button(taskFrame, text= "Add New Task")##default to selected day
editButton = Button(taskFrame, text= "Edit Selected Task") ##if not selected, "please select a task"
removeButton = Button(taskFrame, text= "Remove Selected Task")##same as above and also "Are you sure?"
addButton.pack()
editButton.pack()
removeButton.pack()
settingsButton=Button(taskFrame, text="User Settings")
settingsButton.pack()

taskFrame.pack()

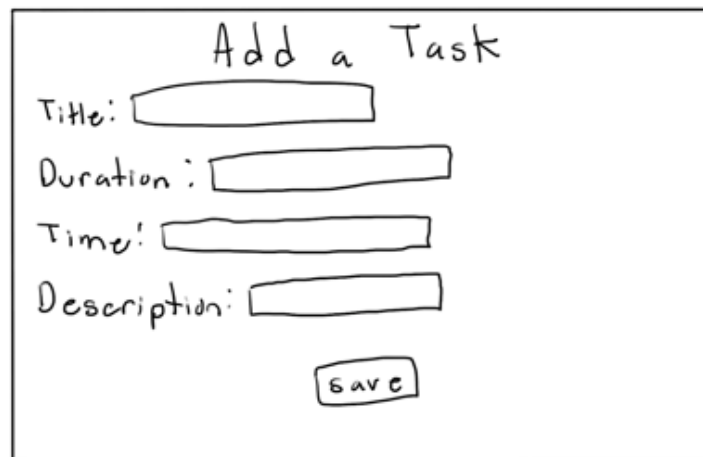
```

## 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.

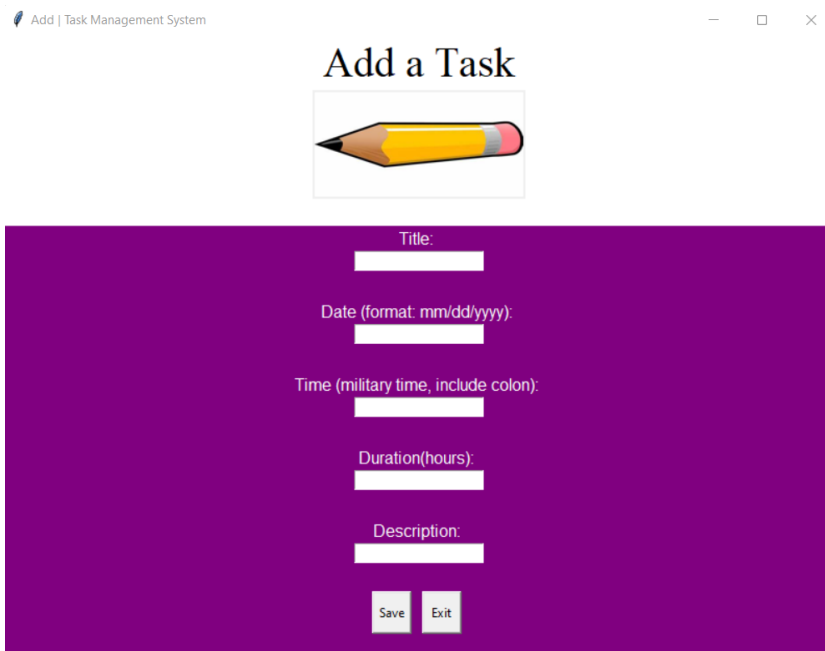
#### Design Idea:



Hand-drawn design idea for the 'Add a Task' page. It shows a title 'Add a Task' at the top, followed by four input fields labeled 'Title:', 'Duration:', 'Time:', and 'Description:'. A 'Save' button is at the bottom right.

Figures 13  
& 14: Add  
Page Layout  
Designs

#### Implementation of Design:



Implementation of the 'Add a Task' page. It shows a window titled 'Add | Task Management System' with a pencil icon. Below the icon is a purple box containing the form. The form has a title 'Add a Task' and a pencil icon. It includes input fields for 'Title:', 'Date (format: mm/dd/yyyy):', 'Time (military time, include colon):', 'Duration(hours):', and 'Description:'. At the bottom are 'Save' and 'Exit' buttons.

# Graphical User Interface Design

## *Add Page*

### Python Code for the Design Implementation:

```
import tkinter as tk

# Creating the adujstable window
window = tk.Tk()
window.title('Add | Task Management System')
window.geometry('500x1000')
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 = 'Add a Task', fg = 'Black', bg = 'white', font = ('Times New Roman', 30))
title.place(x = 300, y = 25)

# Labels
title = tk.Label(Body, text = 'Title: ', fg = 'white', bg = 'purple', font = ('Bold', 12))
title.place(x = 150, y = 50)
time = tk.Label(Body, text = 'Time: ', fg = 'white', bg = 'purple', font = ('Bold', 12))
time.place(x = 150, y = 80)
duration = tk.Label(Body, text = 'Duration: ', fg = 'white', bg = 'purple', font = ('Bold', 12))
duration.place(x = 150, y = 110)
description = tk.Label(Body, text = 'Description: ', fg = 'white', bg = 'purple', font = ('Bold', 12))
description.place(x = 150, y = 140)

# Entries
titleEnt = tk.Entry(Body, width = 70)
titleEnt.place(x = 195, y = 50)
timeEnt = tk.Entry(Body, width = 69)
timeEnt.place(x = 200, y = 80)
durationEnt = tk.Entry(Body, width = 65)
durationEnt.place(x = 225, y = 110)
descriptionEnt = tk.Entry(Body, width = 62)
descriptionEnt.place(x = 245, y = 140)

# Button
save = tk.Button(Body, text = 'Save', width = 4, height = 2).place(x = 350, y = 180)
exitBut = tk.Button(Body, text = 'Exit', width = 4, height = 2).place(x = 400, y = 180)

window.mainloop()
```

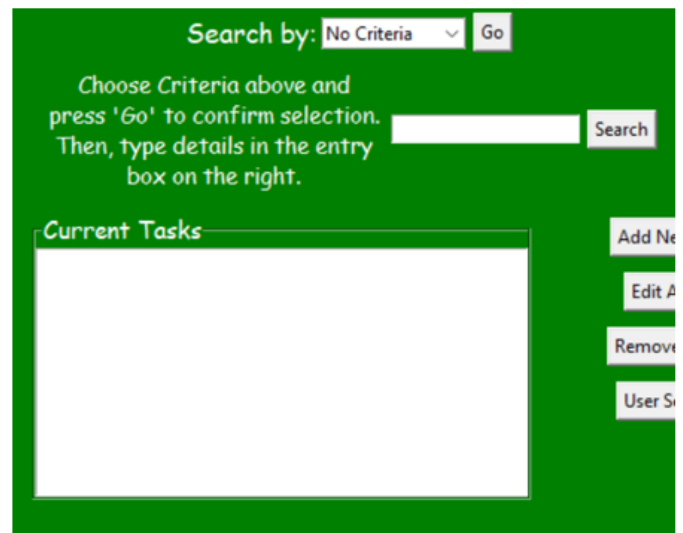
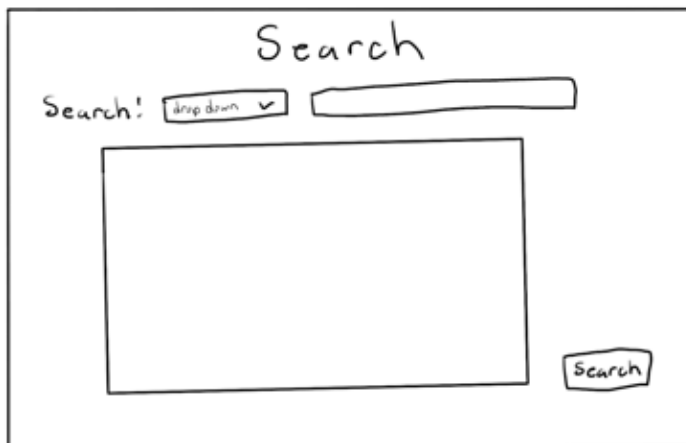
## Graphical User Interface Design

### *Search Page*

The design idea for the search page was to have it be a separate page like all of the previous interface designs. There would be a drop-down box and an entry for a user to enter their criteria, a button to press search, and a big text box in the middle for the output. In the implementation, the search feature is built into the main page. There is still a drop-down and go button like in the design, but the tasks are to output in the current tasks box instead.

#### Implementation of Design:

##### Design Idea:



*Figures 15 & 16: Search Page Layout Designs*

# Graphical User Interface Design

## *Search Page*

### Python Code for the Design Implementation:

```
##Frame for search function
searchFrame = Frame(master=window, bg = 'green')
searchLabel=Label(searchFrame, text="Search by:", bg='Green',
                  fg='white', font=("Comic Sans MS", 10))
searchLabel.pack(side=LEFT)

searchByCombo = Combobox(searchFrame, width=12, state='readonly')
searchByCombo['values']= ("No Criteria", "Month", "Day", "Year", "Time",
                        "Title", "Duration", "Description")
searchByCombo.set("No Criteria Set")

searchByCombo.current(0)
searchByCombo.pack(side=LEFT, pady=10)

goButtonSearch = Button(searchFrame, text="Go")
goButtonSearch.pack(side=LEFT)
searchFrame.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')

##if tasks exist for that day/search. else, label says: "There are no tasks scheduled
task = Radiobutton(taskLabelFrame, text="placeholder") ## has problem: can't deselect,
task.pack()
taskLabelFrame.pack(side=LEFT, padx=70)
```



## 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:**

*Figures 17  
& 18: Edit  
Page Layout  
Designs*

A hand-drawn sketch of a web page titled "Edit a Task". Below the title are four input fields labeled "Title:", "Duration:", "Time:", and "Description:". At the bottom right are two buttons labeled "save edit" and "remove".

#### **Implementation of Design:**

A screenshot of the implemented "Edit a Task" page. The page has a purple background. At the top, there is a header bar with the text "Edit | Task Management System" on the left and window control icons on the right. Below the header, the title "Edit a Task" is displayed next to a close button (X) and a checkmark button. The main content area contains a form with the following elements: a text input field for "Title of the task you would like to edit:", a section header "New Task Details:", and four input fields for "Date:", "Time:", "Duration:", and "Description:". At the bottom of the form are two buttons labeled "Save Edit" and "Exit".

# Graphical User Interface Design

## *Edit Page*

### Python Code for the Design Implementation:

```
import tkinter as tk

# Creating the adjustable window
window = tk.Tk()
window.title('Edit | Task Management System')
window.geometry('500x1000')
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 = 300, y = 25)

# Labels
title = tk.Label(Body, text = 'Title: ', fg = 'white', bg = 'purple', font = ('Bold', 12))
title.place(x = 150, y = 50)
time = tk.Label(Body, text = 'Time: ', fg = 'white', bg = 'purple', font = ('Bold', 12))
time.place(x = 150, y = 80)
duration = tk.Label(Body, text = 'Duration: ', fg = 'white', bg = 'purple', font = ('Bold', 12))
duration.place(x = 150, y = 110)
description = tk.Label(Body, text = 'Description: ', fg = 'white', bg = 'purple', font = ('Bold', 12))
description.place(x = 150, y = 140)

# Entries
titleEnt = tk.Entry(Body, width = 70)
titleEnt.place(x = 195, y = 50)
timeEnt = tk.Entry(Body, width = 69)
timeEnt.place(x = 200, y = 80)
durationEnt = tk.Entry(Body, width = 65)
durationEnt.place(x = 225, y = 110)
descriptionEnt = tk.Entry(Body, width = 62)
descriptionEnt.place(x = 245, y = 140)

# Button
save = tk.Button(Body, text = 'Save Edit', width = 6, height = 2).place(x = 350, y = 180)
exitBut = tk.Button(Body, text = 'Exit', width = 4, height = 2).place(x = 410, y = 180)

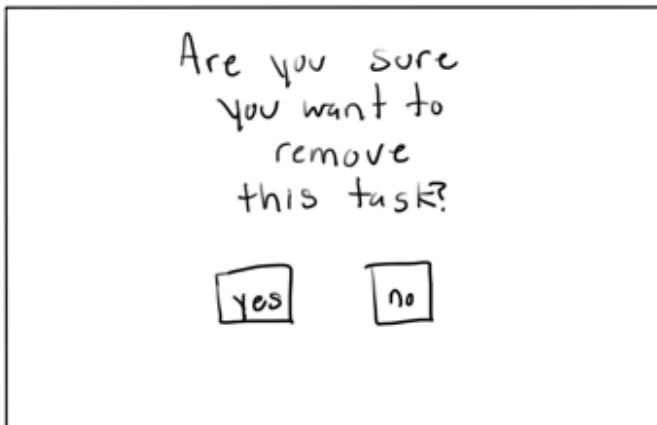
window.mainloop()
```

## 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 19 & 20: Remove Page Layout Designs*

# Graphical User Interface Design

## *Remove Page*

### Python Code for the Desgin Implementation:

```
import tkinter as tk

# Creating the window
window = tk.Tk()
window.title('Remove a Task')
window.geometry('500x300')
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.Frame(window, bg = 'white', height = '100', width = '300')
task.place(x = 93, y = 90)

# 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 = 'Are you sure you would like to remove the task you selected?', fg = 'black', bg = 'lightgray', font = ('Times New Roman', 10))
text.place(x = 75, y = 50)

# Buttons
yes = tk.Button(frame, text = 'Yes', bg = 'green', fg = 'black') #, command =
yes.place(x = 200, y = 220)
no = tk.Button(frame, text = 'No', bg = 'red', fg = 'black') #, command =
no.place(x = 250, y = 220)

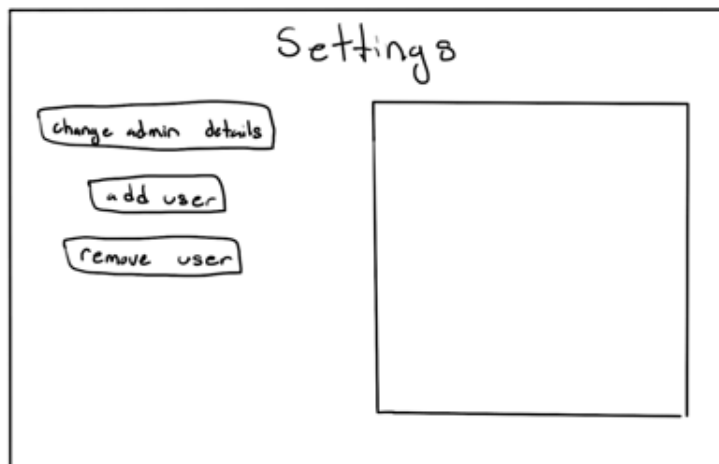
window.mainloop()
```

## Graphical User Interface Design

### *Settings Page*

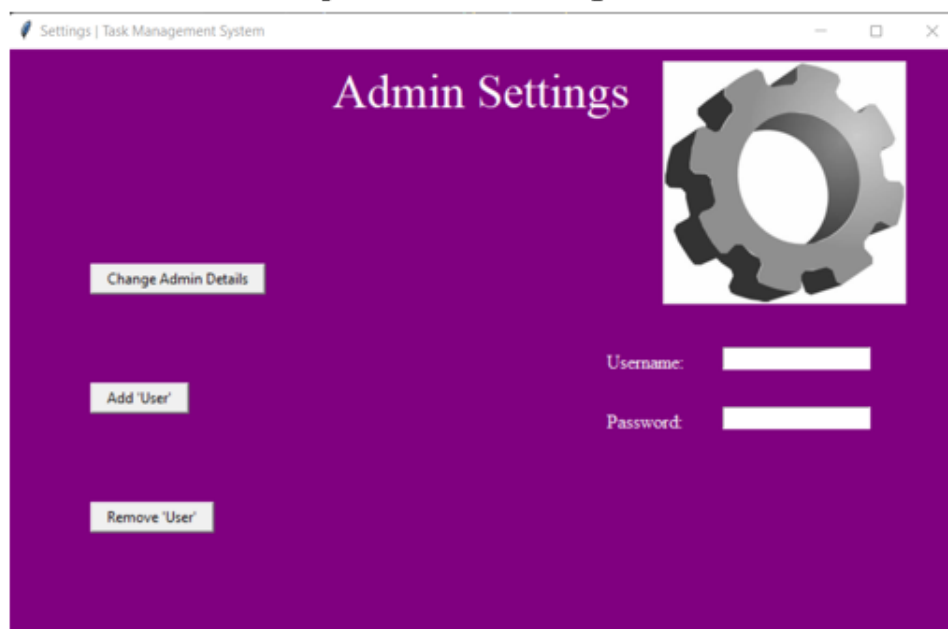
The settings page interface design has an identifying title of Settings at the top and three buttons of change admin details, add user, and remove user. These interface design ideas are still the same in the implementation. The implementation also includes username and password entry boxes as well as an image.

#### Design Idea:



*Figures 21 &  
22: Settings  
Page Layout  
Designs*

#### Implementation of Design:



# Graphical User Interface Design

## *Settings Page*

### Python Code for the Design Implementation:

```
import tkinter as tk

window=tk.Tk()
window.title('Settings')
window.geometry('1000x600')

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

titleLab=tk.Label(window,text="Admin Settings",bg='purple')
titleLab.place(x=450,y=10)

usernLab=tk.Label(text="Username")
usernLab.place(x=600,y=250)

usernEnt=tk.Entry()
usernEnt.place(x=700,y=250)

passLab=tk.Label(text="Password")
passLab.place(x=600,y=300)

passEnt=tk.Entry()
passEnt.place(x=700,y=300)

chgpasBut=tk.Button(text='Change Admin Details')#...command=addition)
chgpasBut.place(x=170,y=180)

adduserBut=tk.Button(text="Add 'User'")#...command=add...
adduserBut.place(x=170,y=280)

chgpasBut=tk.Button(text="Remove 'User'")#...
chgpasBut.place(x=170,y=380)

window.mainloop()

button = tk.Button(
    text="Click me!",
    width=25,
    height=5,
    bg="blue",
    fg="yellow")
```

## 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*

## Data Storage

There are three separate csv files to store all of the necessary information of the Task Manhement System. All three csvs retain their information even after the program is closed so a users' task data is not lost. The first csv, called userLoginInfo, stores the username, password, and admin status for each user. The second csv, called CurrentUser, stores a single row with the username of the current user that is logged in. The third csv, taskList, stores task information such as title, date, time, duration, and description by username. We stored and read information from these csvs using the code from figure 24 below.

### Reading from a CSV:

```
with open ('CurrentUser.csv', 'r') as file:
    csvReader = csv.reader(file)
    for singleRow in csvReader:
        U = singleRow[0]
```

### Writing into a CSV:

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

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

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

*Figure 23: Page Connections Examples*



## 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](https://www.tutorialspoint.com/python/tk_relief.htm)

[Python GUI examples \(Tkinter Tutorial\) - Like](#)

[Geekshttps://pythonguides.com/python-tkinter-radiobutton/](https://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://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/>