# Project Name:

# Alarm Clock with GUI

Name: YEDUVAKA LIKHITHA

Designation: Python Development Intern

Intern ID:CC44573

Batch: August 2023

Mail ID: 20981a4914@raghuenggcollege.in

# 

# Context Introduction and Usage:

An Alarm Clock with GUI is a software application designed to provide users with a visually interactive way to set and manage alarms. Unlike traditional alarm clocks, which may have limited functionalities, a GUI-based Alarm Clock offers a more user-friendly experience, allowing users to set multiple alarms, customize alarm sounds, snooze options, and more.

The primary purpose of an Alarm Clock with GUI is to provide an intuitive interface for users to set alarms and receive timely reminders for various tasks, events, or wake-up calls. The graphical user interface (GUI) enhances the user experience by making it easy to interact with the alarm clock's features.

# Objective:

The objective of this intern project is to develop a simple Alarm Clock application with a graphical user interface using Python and a GUI library (e.g., Tkinter) to provide an intuitive and user-friendly experience for setting and managing alarms.

# Software Requirements:

1. Python: The project requires Python 3.x to be installed on the development environment.

2. GUI Library (Tkinter): Utilize the Tkinter library to create the graphical user interface for the Alarm Clock application.

3. IDE (Integrated Development Environment): The intern can choose one of the following IDEs for development:

-Visual Studio Code

- PyCharm

- Thonny

4. Alarm Management: Implement the alarm management system to store and handle the alarms with their respective settings. Use data structures to store alarm details like time, days to repeat, alarm sound, and label

5. Alarm Triggering: The application should be able to trigger alarms at the specified time and date, even when the application is not running.

6. Alarm Sound: Play a selected sound or music file when the alarm is triggered.

7. Alarm Actions: Provide options to display a message or execute a user-defined action when the alarm is triggered.

8. Snooze Functionality: Implement a snooze option that allows users to postpone the alarm for a specified duration.

9. Alarm Persistence: Ensure that the alarms persist between application sessions, so users don't lose their configured alarms after closing the application.

10. User Interface (UI):

- Create an aesthetically pleasing and intuitive graphical user interface using Tkinter widgets.

- Design a user-friendly layout that allows users to set alarms easily with fields for time, repeating days, sound selection, label, and snooze duration.

- Implement buttons for setting new alarms, deleting alarms, and managing alarm actions.

11. Error Handling: Implement error handling mechanisms to handle invalid user inputs, file access issues, or other unexpected errors gracefully.

12. Testing: Develop a testing strategy to ensure the application functions correctly under various scenarios, including edge cases and alarm triggering behavior.

13. Documentation: Prepare clear and concise documentation explaining the project, how to install dependencies, how to run the application, and any other relevant information for future reference.

# Project Prerequisities:

This project requires good knowledge of Python and GUI (Graphic User Interface). Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. All the modules used need not be downloaded beforehand like the other libraries like NumPy, thus this project will be user friendly and accessible in any virtual environment used for python programming.

# Project File Structure:

The steps to build an Alarm Clock program in Python:

* Importing all the libraries and modules required
* Putting forward a while loop which takes the argument of the time, the user wants to set the alarm on and automatically breaks when the time is up, with sound
* Create a display window for user input.

# Step1: Import all the necessary Libraries and modules:

#Importing all the necessary libraries to form the alarm clock:

from tkinter import \*

import datetime

import time

import winsound

# Explanation:

* Helps us to create a dialog box with any information that we want to provide or get from the users.
* [**Datetime**](https://docs.python.org/3/library/datetime.html) and **time** modules in python help us to work with the dates and time of the current day when the user is operating python and to manipulate it too.
* **Winsound** module provides access to the basic sound playing machinery provided by Windows platforms. This is useful to generate the sound immediately when a function is called.

# Step2: Create a while Loop

def alarm(set\_alarm\_timer):

while True:

time.sleep(1)

current\_time = datetime.datetime.now()

now = current\_time.strftime("%H:%M:%S")

date = current\_time.strftime("%d/%m/%Y")

print("The Set Date is:",date)

print(now)

if now == set\_alarm\_timer:

print("Time to Wake up")

winsound.PlaySound("sound.wav",winsound.SND\_ASYNC)

break

def actual\_time():

set\_alarm\_timer = f"{hour.get()}:{min.get()}:{sec.get()}"

alarm(set\_alarm\_timer)

# Explanation:

* Define a function named as **alarm()** which takes the argument of (**set\_alarm\_timer**).It contains a while loop with a Boolean function True which makes the program automatic to work.
* **time.sleep(1)** halts the execution of the further commands given until we get the time value from the user later in the code and returns the background thread of the clock time going on at a regular interval.
* Get the current time using **current\_time** which takes the argument of **datetime.datetime.now()**.
* now is used to print the time and date is used to print the current date by **string conversion** using **strftime()**.
* Define another function here named **actual\_time()** which takes in the user value for setting the alarm in the string format. The same argument of (**set\_alarm\_timer**) as alarm before to execute the while loop which we further use while making GUI.
* If loop suggests that if the user input time **set\_alarm\_timer** matches with the while loop ongoing time now, the message is printed as” **Time to Wake up**”.
* **winsound.SND\_ASYNC** plays the system generated sound as soon the condition satisfies, acting as a reminder for the alarm clock.

# Step3: Creating GUI using tkinter:

clock = Tk()

clock.title("DataFlair Alarm Clock")

clock.geometry("400x200")

time\_format=Label(clock, text= "Enter time in 24 hour format!", fg="red",bg="black",font="Arial").place(x=60,y=120)

addTime = Label(clock,text = "Hour Min Sec",font=60).place(x = 110)

setYourAlarm = Label(clock,text = "When to wake you up",fg="blue",relief = "solid",font=("Helevetica",7,"bold")).place(x=0, y=29)

# The Variables we require to set the alarm(initialization):

hour = StringVar()

min = StringVar()

sec = StringVar()

#Time required to set the alarm clock:

hourTime= Entry(clock,textvariable = hour,bg = "pink",width = 15).place(x=110,y=30)

minTime= Entry(clock,textvariable = min,bg = "pink",width = 15).place(x=150,y=30)

secTime = Entry(clock,textvariable = sec,bg = "pink",width = 15).place(x=200,y=30)

#To take the time input by user:

submit = Button(clock,text = "Set Alarm",fg="red",width = 10,command = actual\_time).place(x =110,y=70)

clock.mainloop()

#Execution of the window.

# Explanation:

* To Initialize tkinter, we pass a command under the name clock as **Tk()**.
* The dialog box has the title as DataFlair Alarm Clock with a geometry of (400\*200). We pass on the heading to mention the time format for 24 hours using **time\_format**.
* The second heading is given above the user input boxes for the labeling to be “Hour Min Sec” using **addTime**.
* Just to make the dialog box look funkier, adding another label as “**when to wake you up**” using **setYourAlarm**.
* As we have already converted the current time in the string before (actual time), the variables we initialize for the user input dialog boxes are in **StringVar()**.
* Finally make the input boxes such as **hourTime**, **minTime,** and **secTime** which takes the entry of the time the user wants to set the alarm on in 24-hour format.
* Submit takes the command of the defined function **actual\_time** and executes the clock as it acts as a set button to start the program.
* **Clock.mainloop()** is the basic and the last command was given to compile all the previous commands with their basic settings of color, font, width, axis, etc. and displays the window as soon as the program is run.

# Source Code:

#Importing all the necessary libraries to form the alarm clock:

from tkinter import \*

import datetime

import time

import winsound

def alarm(set\_alarm\_timer):

while True:

time.sleep(1)

current\_time = datetime.datetime.now()

now = current\_time.strftime("%H:%M:%S")

date = current\_time.strftime("%d/%m/%Y")

print("The Set Date is:",date)

print(now)

if now == set\_alarm\_timer:

print("Time to Wake up")

winsound.PlaySound("sound.wav",winsound.SND\_ASYNC)

break

def actual\_time():

set\_alarm\_timer = f"{hour.get()}:{min.get()}:{sec.get()}"

alarm(set\_alarm\_timer)

clock = Tk()

clock.title("DataFlair Alarm Clock")

clock.iconbitmap(r"dataflair-logo.ico")

clock.geometry("400x200")

time\_format=Label(clock, text= "Enter time in 24 hour format!", fg="red",bg="black",font="Arial").place(x=60,y=120)

addTime = Label(clock,text = "Hour Min Sec",font=60).place(x = 110)

setYourAlarm = Label(clock,text = "When to wake you up",fg="blue",relief = "solid",font=("Helevetica",7,"bold")).place(x=0, y=29)

# The Variables we require to set the alarm(initialization):

hour = StringVar()

min = StringVar()

sec = StringVar()

#Time required to set the alarm clock:

hourTime= Entry(clock,textvariable = hour,bg = "pink",width = 15).place(x=110,y=30)

minTime= Entry(clock,textvariable = min,bg = "pink",width = 15).place(x=150,y=30)

secTime = Entry(clock,textvariable = sec,bg = "pink",width = 15).place(x=200,y=30)

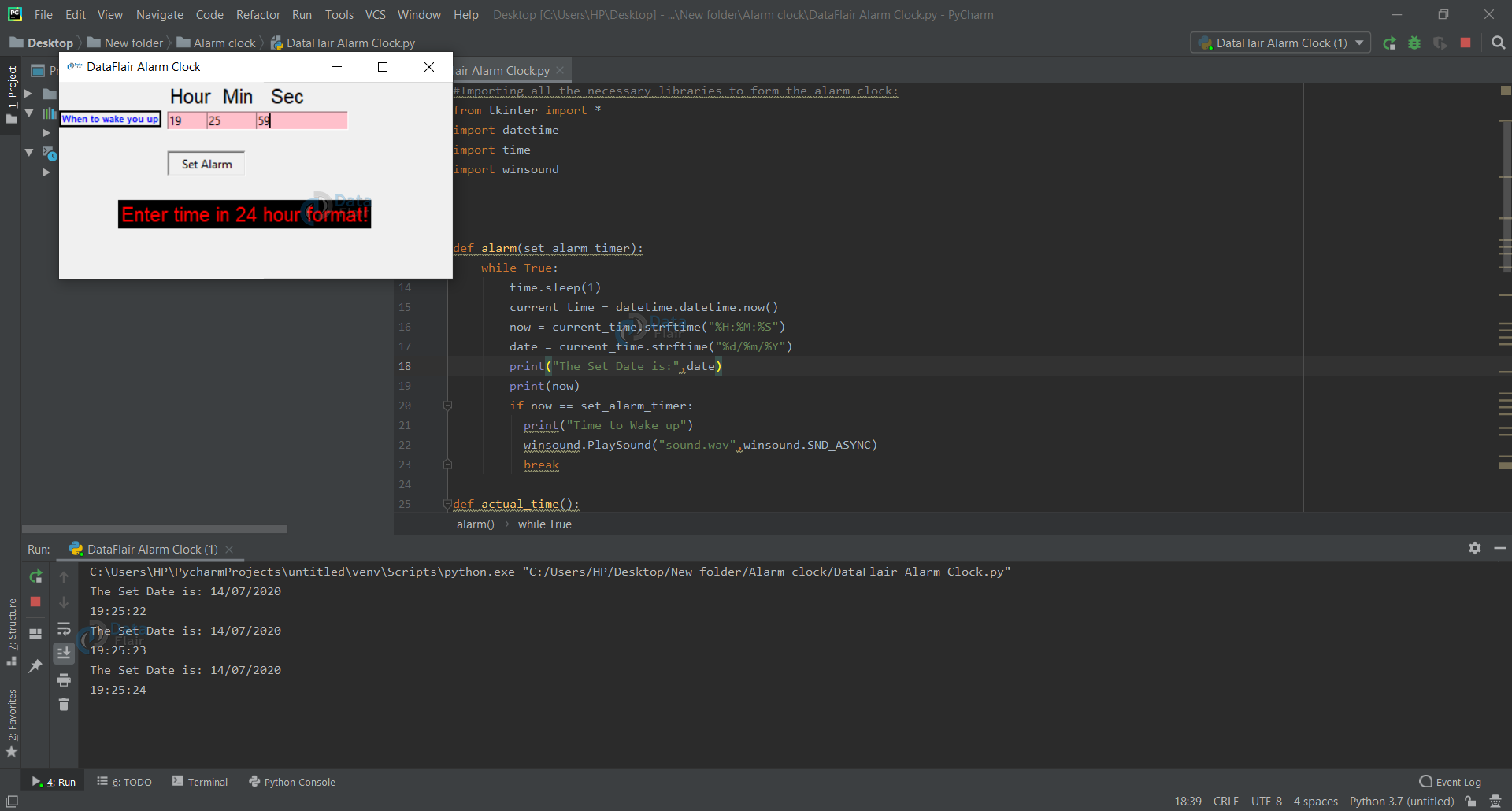
#To take the time input by user:

submit = Button(clock,text = "Set Alarm",fg="red",width = 10,command = actual\_time).place(x =110,y=70)

clock.mainloop()

#Execution of the window.

# OUTPUT:



# Summary:

With this project in Python, we have successfully made the Alarm Clock. We used the popular GUI library for rendering graphics on a display window. We learned how to extract the current time from the computer and to use it for manipulation using the DateTime library. This way we can set an alarm in the computer interface using python programming which rings with the default machine sound for Windows.

