ALARM CLOCK USING PYTHON GUI

PROJECT REPORT

By

Isha Anand(RA2311026030008)
Aviral Jain(RA2311026030035)
Vijaya Laxmi Ruhela(RA2311026030051)
Jiya Arya(RA2311026030056)

Under the guidance of

Ms. Bhawna Upadhayay

In partial fulfilment for the Course

of

21CSC203P – ADVANCED PROGRAMMING

PRACTICE in COMPUTER SCIENCE AND

ENGINEERING

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

DELHI-NCR CAMPUS

October-2024

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report for the course **21CSC203P ADVANCED PROGRAMMING PRACTICE** entitled in "**ALARM CLOCK USING PYTHON GUI**" is the bonafide work of **Isha Anand(RA2311026030008),**

Aviral Jain(RA2311026030035),Vijaya Laxmi Ruhela(RA2311026030051)

Jiya Arya(RA2311026030056) who carried out the work under my supervision.

SIGNATURE

Ms. Bhawna Upadhayay

(Assistant Professor)
Dept. of Computer Science & Engineering

SIGNATURE

Dr. Avneesh Vashistha

HEAD OF THE DEPARTMENT
Dept. of Computer Science & Engineering

ABSTRACT

This alarm clock project, developed using Python and Tkinter for the graphical user interface (GUI), offers a functional and user-friendly tool to set and manage alarms. The application allows users to customize multiple alarms with personalized labels, sounds, and snooze options. It supports both 12-hour and 24-hour time formats, providing flexibility for global users, and displays the current time in real-time.

Key features include an intuitive interface where users can easily add, edit, or delete alarms, set specific days for recurring alarms, and choose custom sound notifications from a list of pre-installed tones or personal audio files. The snooze functionality allows users to delay alarms for a user-defined interval, ensuring they don't miss important tasks while allowing flexibility.

The project highlights Python's capabilities in handling real-time operations, managing system time, and interacting with external libraries for enhanced functionality. Tkinter, as the GUI framework, provides a simple yet effective way to design the user interface, while additional Python libraries like time and playsound are used to manage time-based events and audio playback.

This project serves as a hands-on demonstration of Python's practical use in building functional desktop applications and introduces concepts like event handling, threading, and time management.

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my teacher **Ms. Bhawna Upadhayay** as well as our HOD of CSE **Dr. Avneesh Vashistha** who gave me the golden opportunity to do this wonderful project report on **ALARM CLOCK USING PYTHON GUI** which also helped me in doing a lot of research and came to know about so many new things. I am really thankful to them. Secondly, I would also like to thank my parents and friends who helped me a lot in finishing this project within the limited time.

ROLES

Name	Role
Isha Anand	**Backend** Responsibilities: Develop the core functionality of the alarm clock using Python, focusing on scheduling and triggering alarms. Manage time-related tasks, including countdowns and notifications when alarms go off. Implement data storage solutions (like file handling) for saving user preferences and alarm settings.
Aviral Jain	**Idea** Responsibilities: Brainstorm and define the core features and functionality of the alarm clock (e.g., alarm settings, snooze options, themes). Develop a unique selling proposition that differentiates the application from existing ones. Outline the user experience flow, considering how users will interact with the application
Vijaya Laxmi Ruhela	**Frontend** Responsibilities: Design and implement the user interface using Tkinter, ensuring it is visually appealing and easy to navigate. Create widgets for setting alarms, displaying time, and adjusting settings. Ensure the application is responsive and user-friendly, with clear feedback for user interactions.
Jiya Arya	**Research** Responsibilities: Investigate existing alarm clock applications to identify their strengths, weaknesses, and user reviews. Study user needs and preferences through surveys or interviews to understand desired features. Explore best practices in UI/UX design, particularly for alarm clock applications, to inform the frontend design.

TABLE OF CONTENTS

INTRODUCTION

An alarm clock plays a crucial role in helping individuals manage their daily routines, ensuring punctuality and productivity. In today's digital age, the functionality of an alarm clock has expanded beyond simple timekeeping to include multiple customization options, making it a vital tool for everyday life. This project aims to develop a Python-based alarm clock application using a graphical user interface (GUI) built with Tkinter, one of Python's most popular libraries for creating desktop applications.

The main goal of this project is to provide users with a simple yet highly functional alarm clock system that they can easily interact with. Users will be able to set multiple alarms, each with customizable labels and sounds, offering flexibility for different time management needs, whether for daily reminders, wake-up alarms, or one-time events. The application also includes features such as adjustable snooze settings and the ability to choose between 12-hour and 24-hour time formats.

One of the key components of this project is the real-time event handling and time-based functionality, demonstrating Python's strength in managing real-time data and event-driven operations. Python's `time` and `datetime` libraries are used for accurate timekeeping, while the Tkinter library provides the structure for the GUI, ensuring a seamless user experience.

The alarm clock project showcases Python's versatility by integrating sound playback for alarm notifications, which can be customized by users either through pre-installed tones or their own audio files. Additionally, the snooze functionality allows users to pause alarms for a user-defined interval, ensuring flexibility in daily routines.

This project not only highlights Python's ability to create practical and efficient desktop applications but also introduces concepts of event-driven programming, threading, and time management. For individuals seeking to build a personalized time management tool or expand their knowledge of GUI programming, this project provides a hands-on approach to understanding how Python can be used to develop interactive, real-world applications. Through this alarm clock project, users and developers alike will gain valuable insights into Python's real-time capabilities and how it can be leveraged to solve everyday problems efficiently.

WORKING

The alarm clock application has several real-world applications, making it a versatile and practical tool for everyday use. Below are some of the key scenarios where the alarm clock can be especially useful:

1**. Daily Reminders**:

The alarm clock can be used to set alarms for important tasks, appointments, or meetings. For users with busy schedules, this feature serves as a helpful reminder system. Whether it's for daily commitments like picking up children from school, attending a workout session, or work-related tasks, users can rely on the clock to stay organized. This capability enhances productivity by ensuring that crucial events are not missed.

2. **Time Management:**

The application can also be utilized as a tool for effective time management. Users can organize their work or study sessions by setting alarms to mark different periods of the day. For example, someone using the Pomodoro technique (where tasks are divided into short intervals) can use the alarm clock to track work and break times. This time-bound approach helps people stay focused, manage distractions, and optimize productivity throughout the day.

3. **Wake-Up Alerts:**

In its simplest form, the alarm clock serves as a wake-up alarm for specific times of the day. Users can set daily alarms to help them wake up on time, ensuring that they start their day punctually. The ability to customize alarms by selecting different sounds or snooze durations adds flexibility, making it a personalized tool for morning routines.

Beyond practical applications, the project demonstrates how Python can be applied to build functional desktop utilities with minimal setup and coding effort. The project leverages Tkinter for the graphical interface, along with libraries like `time` and `playsound` for time handling and sound playback. This combination shows the strength of Python in handling real-time operations and user interaction in a desktop application environment.One of the highlights of the alarm clock project is its simplicity and customization. With a lightweight structure, it's ideal for users who want a basic, easy-to-use application that fulfills the essential functions of an alarm clock. Additionally, for developers, the project is highly extendable. They can modify the code to add new features, such as recurring alarms on specific days, integration with calendar apps, or even syncing alarms across multiple devices.

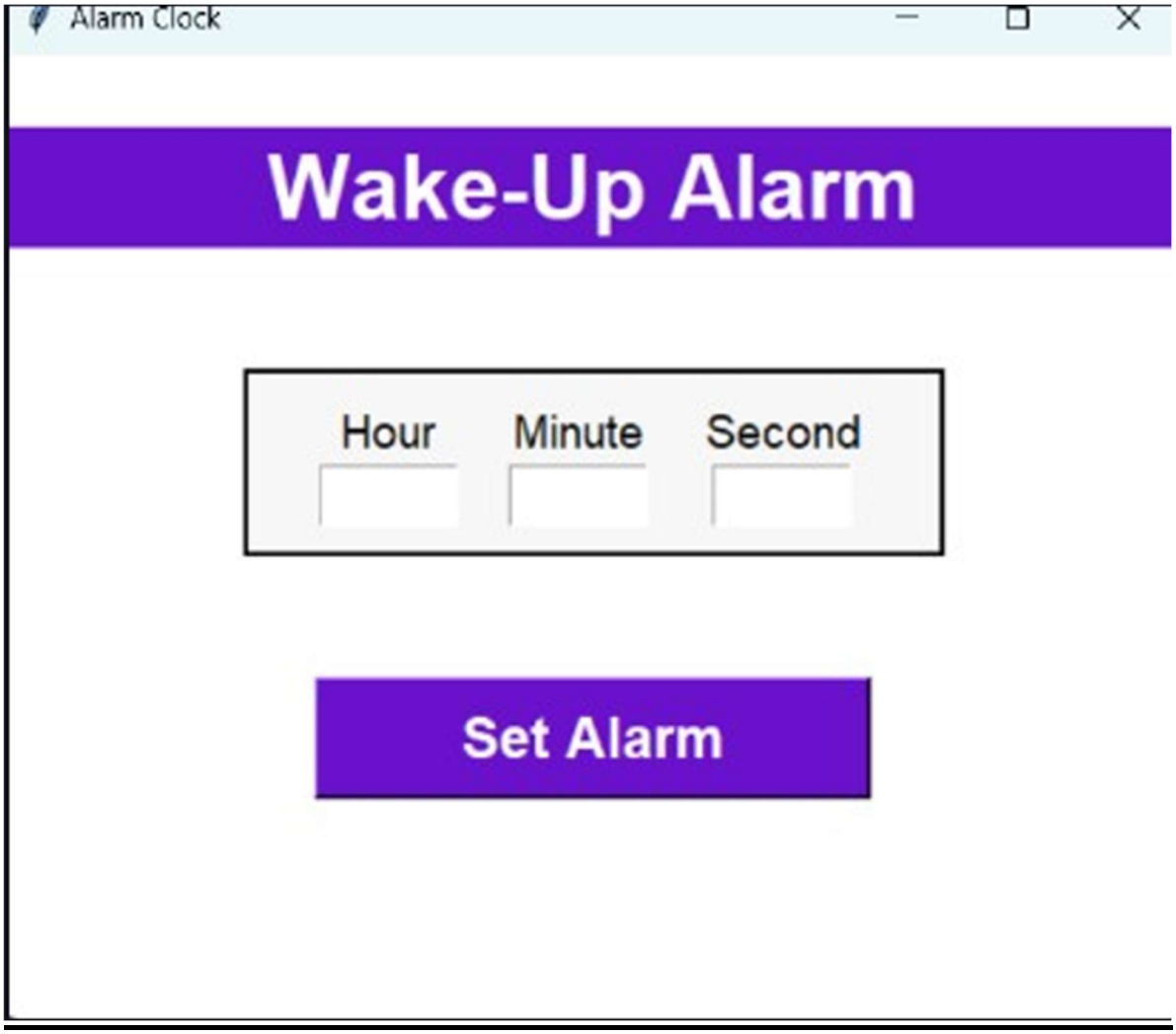
SOURCE CODE

```
import tkinter as tk
from tkinter import messagebox
import time
import threading
import winsound  # Built-in sound module for Windows

# Function to set the alarm
def set_alarm():
    alarm_time = f"{hour.get()}:{minute.get()}:{second.get()}"
    messagebox.showinfo("Alarm Set", f"Alarm is set for {alarm_time}")
    while True:
        current_time = time.strftime("%H:%M:%S")
        if current_time == alarm_time:
            ring_alarm()
            break
        time.sleep(1)

# Function to ring the alarm
def ring_alarm():
    label_alarm_status.config(text="Time's Up!", fg="white", bg="#ff4747")
    root.update()

    # Play sound using winsound for Windows users
    winsound.PlaySound("alarm_sound.wav", winsound.SND_FILENAME)

    messagebox.showinfo("Alarm", "Time to wake up!")

# Function to start the alarm in a separate thread
```

```
def start_alarm_thread():
    t = threading.Thread(target=set_alarm)
    t.start()

# Initialize Tkinter window
root = tk.Tk()
root.title("Alarm Clock")
root.geometry("500x400")
root.config(bg="#ffffff")

# Title Label with custom font and gradient background
label_title = tk.Label(root, text="Wake-Up Alarm", font=("Helvetica", 28, "bold"), fg="white",
bg="#6a11cb", width=25)
label_title.pack(pady=30)

# Time Input Frame with rounded corners and styled background
frame_time_input = tk.Frame(root, bg="#f7f7f7", bd=2, relief="solid", padx=20, pady=10)
frame_time_input.pack(pady=20)

# Hour input
label_hour = tk.Label(frame_time_input, text="Hour", font=("Helvetica", 14), bg="#f7f7f7")
label_hour.grid(row=0, column=0, padx=10)
hour = tk.Entry(frame_time_input, width=5, font=("Helvetica", 14), justify="center")
hour.grid(row=1, column=0, padx=10)

# Minute input
label_minute = tk.Label(frame_time_input, text="Minute", font=("Helvetica", 14), bg="#f7f7f7")
label_minute.grid(row=0, column=1, padx=10)
minute = tk.Entry(frame_time_input, width=5, font=("Helvetica", 14), justify="center")
minute.grid(row=1, column=1, padx=10)
```

```
# Second input
label_second = tk.Label(frame_time_input, text="Second", font=("Helvetica", 14), bg="#f7f7f7")
label_second.grid(row=0, column=2, padx=10)
second = tk.Entry(frame_time_input, width=5, font=("Helvetica", 14), justify="center")
second.grid(row=1, column=2, padx=10)

# Set Alarm Button with hover effect
def on_enter(e):
    button_set_alarm['bg'] = '#ff6363'

def on_leave(e):
    button_set_alarm['bg'] = '#6a11cb'

button_set_alarm = tk.Button(root, text="Set Alarm", font=("Helvetica", 18, "bold"), bg="#6a11cb",
fg="white", width=15, command=start_alarm_thread)
button_set_alarm.pack(pady=30)
button_set_alarm.bind("<Enter>", on_enter)
button_set_alarm.bind("<Leave>", on_leave)

# Alarm status label
label_alarm_status = tk.Label(root, font=("Helvetica", 14), bg="#ffffff")
label_alarm_status.pack(pady=10)

# Run the app
root.mainloop()
```

OUTPUT



CONCLUSION

In this project, we developed a functional alarm clock application using Python and Tkinter, focusing on creating a user-friendly interface for setting alarms and managing schedules. The application showcases Python's versatility in building practical desktop applications while emphasizing ease of use.

By integrating multithreading, the alarm clock can handle user input and background tasks simultaneously. This ensures a responsive user experience, allowing alarms to trigger without interrupting ongoing interactions. Utilizing Python’s threading capabilities, we maintained smooth operation while users interacted with the interface.

Tkinter proved to be an efficient library for rapid GUI development, particularly for beginners. Its intuitive design enabled me to create various interface elements, such as buttons and entry fields, facilitating a clean and organized layout. This project highlights how Python can be used effectively beyond backend development, creating useful applications with minimal dependencies.

In conclusion, the alarm clock application not only demonstrates the effectiveness of Python for desktop development but also inspires further exploration into more advanced features, such as multiple alarms and sound notifications. This experience has reinforced our

REFERENCES

- https://docs.python.org/3/library/tkinter.html
- https://www.w3schools.com/python/python_reference.asp
- https://www.geeksforgeeks.org/python-gui-tkinter/
- https://www.pythontutorial.net/tkinter/