

Designing & Developing a Digital Alarm Clock in Python

Presented by: AKSHITA KHARE 25BAI10481

Date: 24-11-2025

Project: Python Digital Alarm Clock



Introduction: Crafting a Smart Alarm

Our mission: to design and build an intuitive, user-friendly digital alarm clock using Python. This project will leverage standard Python libraries and explore GUI development.



User-Friendly Design

Focus on intuitive interaction.



Python-Powered

Utilizing core Python strengths.



Full Project Scope

From initial idea to final testing.





Defining the Challenge: Our Problem Statement

The market often lacks simple, customizable alarm clock applications. Many existing solutions are either overly complex, platform-locked, or offer limited personalization.



Lack of Customization

Existing clocks often fall short in offering personalized settings.



Platform Dependency

Many are tied to specific operating systems, limiting accessibility.



Complex Interfaces

Some are too cluttered, making simple tasks difficult for users.

Our goal is to overcome these limitations by developing a straightforward, cross-platform, and highly usable digital alarm clock with a clear graphical user interface.

Functional Requirements: What it Must Do



Set Alarm Time

Users can easily set the alarm in 24-hour format.



Current Time Display

The clock must show the current time, updated every second.



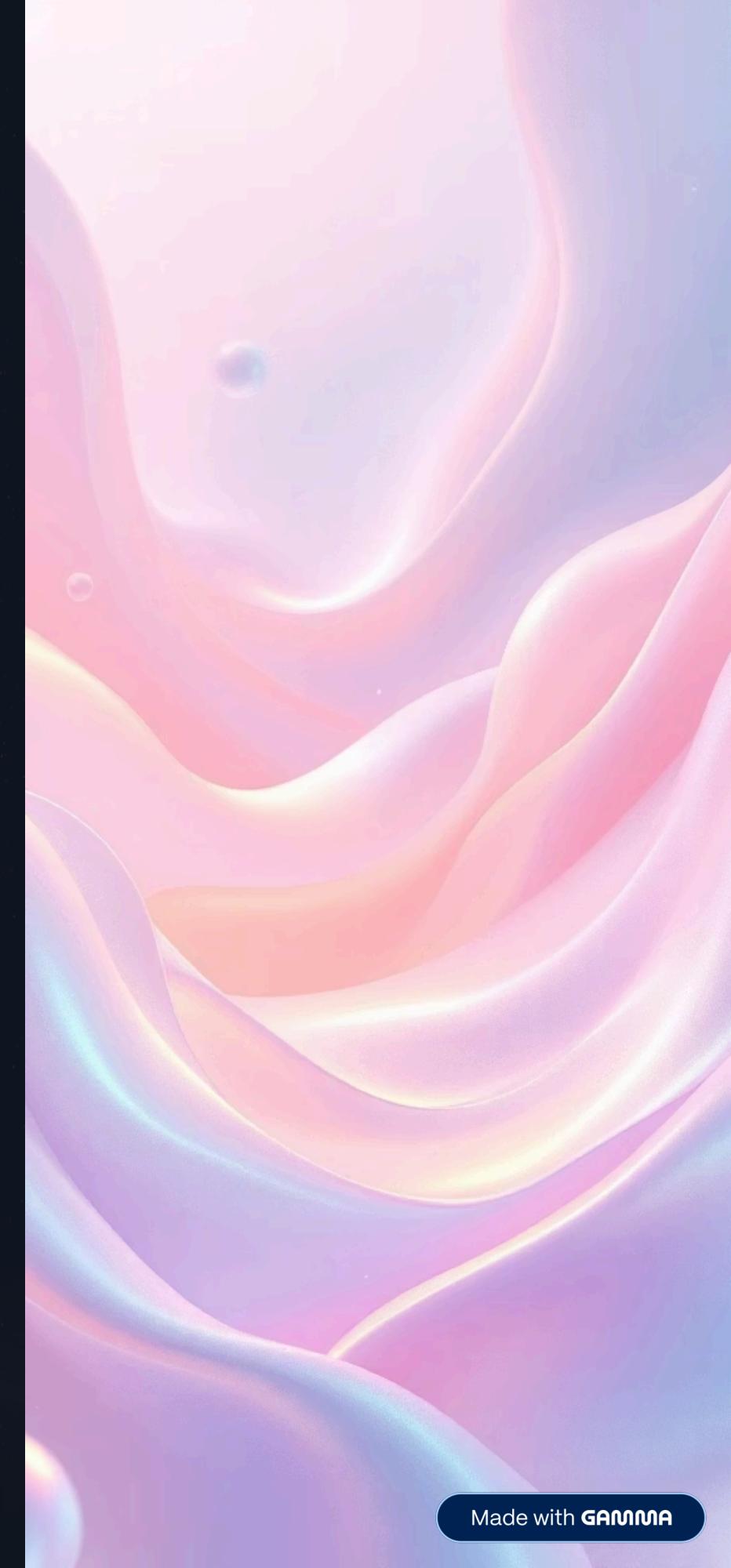
Alarm Sound Playback

A distinct sound will play when the alarm time is reached.



Start/Stop Control

Users need clear buttons to activate and deactivate the alarm.



Non-Functional Requirements: How it Must Perform

Beyond its core functions, our alarm clock must meet key quality attributes to ensure a superior user experience.



Usability

Intuitive GUI with clear labels and controls.



Reliability

Accurate time tracking and consistent alarm triggering.



Performance

Minimal CPU usage to run efficiently in the background.



Portability

Designed to run on Windows and adaptable for other operating systems.



Maintainability

Modular code for easy updates and future enhancements.

Implementation Details: Bringing it to Life with Python

Key Technologies

- **Tkinter:** For creating the graphical user interface.
- **Datetime:** To handle all time-related operations.
- **Time:** For precise delays and loops.
- **Winsound/Playsound:** For playing alarm audio across platforms.

Core Logic

A background loop continuously compares the current system time with the user-set alarm time, activating the alarm when they match.

GUI Elements

Labels display information, OptionMenus allow easy time selection, and Buttons provide control over the alarm state.

Concurrency

Threading ensures the GUI remains responsive while the alarm logic runs independently in the background, preventing freezing.



Testing Approach: Ensuring Reliability

A rigorous testing strategy ensures the alarm clock functions as expected under various conditions, delivering a dependable user experience.



Unit Testing

Individual components like time parsing and sound playback are tested in isolation.



Integration Testing

Verifying that GUI elements correctly interact with backend alarm logic.



Functional Testing

Setting alarms for various times (e.g., immediate, future, cross-day) to confirm correct triggering.



Usability Testing

Gathering user feedback on the interface's intuitiveness and ease of use.

Learnings & Key Takeaways

This project provided valuable insights into Python GUI development, concurrency, and real-time application design.



GUI with Tkinter

Mastered basic and advanced Tkinter widgets and layout management.

Concurrency in Python

Understood the importance and implementation of threading for responsive applications.

Time Management

Gained proficiency in handling date/time objects and comparisons.

Cross-Platform Challenges

Identified and addressed platform-specific dependencies for audio playback.

Future Enhancements: Expanding Possibilities

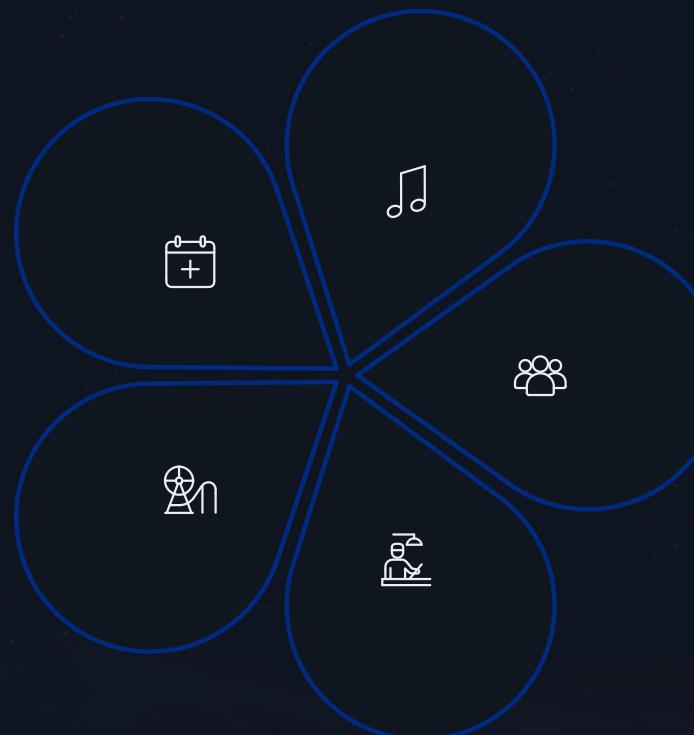
The current alarm clock serves as a solid foundation, with numerous avenues for future development and expanded functionality.

Snooze Feature

Implement a temporary pause for the alarm.

Theming & Skins

Offer customizable visual themes for the GUI.



Custom Alarm Sounds

Allow users to select their own audio files.

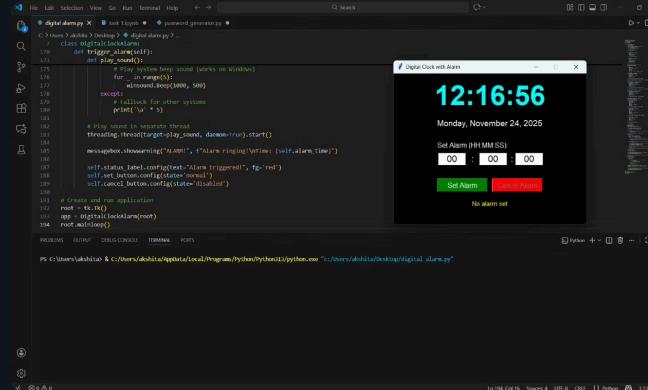
Multiple Alarms

Enable setting and managing several distinct alarms.

OS Integration

Better integration with system notifications and startup.

Screenshots & Results: The Working Prototype

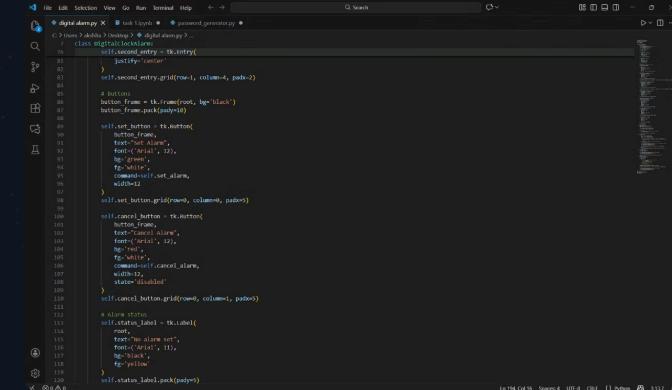


```
File Edit Selection View Go Run Terminal Help ⌘ S Search Q Search C S C

C:\Users\abubka\Desktop\Digital Clock\digital_clock.py
170     def trigger_alarm(self):
171         self.alarm_active = True
172         self.play_sound()
173         for _ in range(10):
174             winsound.Beep(1000, 500)
175         except:
176             print("An error occurred while triggering the alarm")
177             messagebox.showerror("Error", "An error occurred while triggering the alarm")
178             self.set_button.config(state="normal")
179             self.cancel_button.config(state="disabled")
180             self.root.mainloop()
181
182 # Create and run application
183 root = Tk()
184 app = DigitalClock(root)
185 app.root.mainloop()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PS Python + v - i 3137 D
```

PS C:\Users\abubka\Desktop\Digital Clock\digital_clock> python digital_clock.py

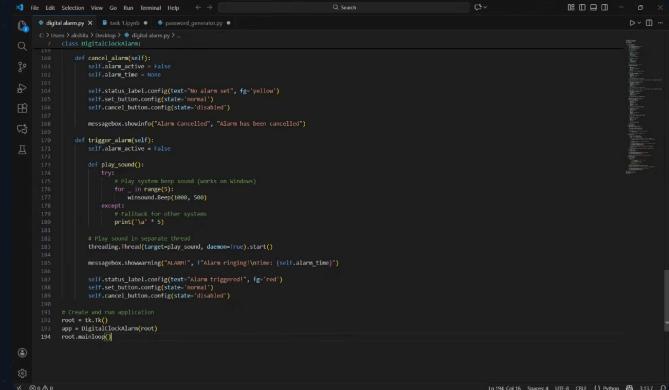


```
File Edit Selection View Go Run Terminal Help ⌘ S Search Q Search C S C

C:\Users\abubka\Desktop\Digital Clock\digital_clock.py
170     self.cancel_alarm(self):
171         self.alarm_active = False
172         self.play_sound()
173         for _ in range(10):
174             winsound.Beep(1000, 500)
175         except:
176             print("An error occurred while cancelling the alarm")
177             messagebox.showerror("Error", "An error occurred while cancelling the alarm")
178             self.set_button.config(state="normal")
179             self.cancel_button.config(state="disabled")
180             self.root.mainloop()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PS Python + v - i 3137 D
```

PS C:\Users\abubka\Desktop\Digital Clock\digital_clock> python digital_clock.py

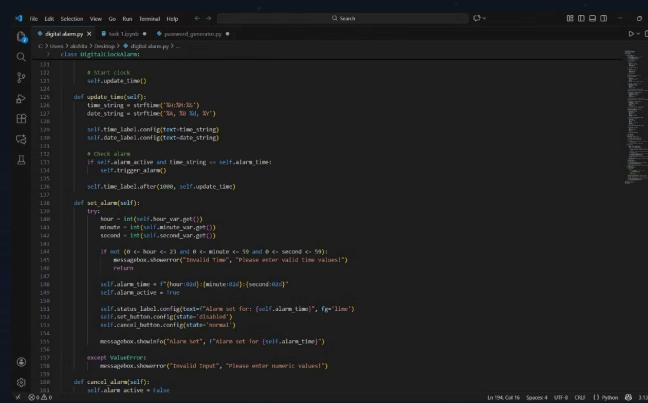


```
File Edit Selection View Go Run Terminal Help ⌘ S Search Q Search C S C

C:\Users\abubka\Desktop\Digital Clock\digital_clock.py
170     def play_sound(self):
171         try:
172             self.play_system_beep_sound()
173             for _ in range(10):
174                 winsound.Beep(1000, 500)
175             except:
176                 print("An error occurred while playing the sound")
177                 messagebox.showerror("Error", "An error occurred while playing the sound")
178                 self.set_button.config(state="normal")
179                 self.cancel_button.config(state="disabled")
180                 self.root.mainloop()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PS Python + v - i 3137 D
```

PS C:\Users\abubka\Desktop\Digital Clock\digital_clock> python digital_clock.py



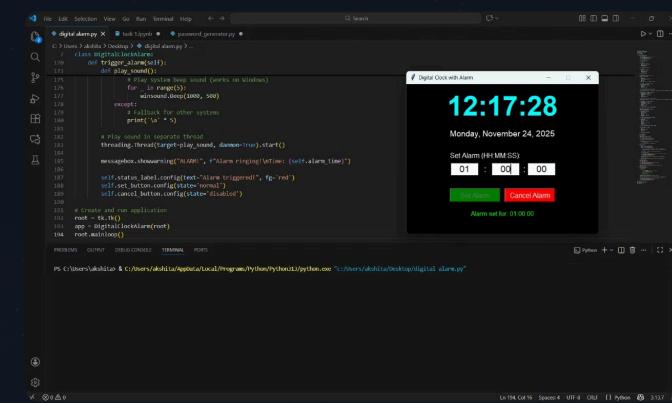
```
File Edit Selection View Go Run Terminal Help ⌘ S Search Q Search C S C

C:\Users\abubka\Desktop\Digital Clock\digital_clock.py
170     def set_alarm(self):
171         self.start_clock()
172         self.update_time()
173
174     def update_time(self):
175         date_string = strftime("%Y-%m-%d %H:%M:%S")
176         date_string += strftime("%S", gmtime())
177         self.time_label.config(text=date_string)
178         self.date_label.config(text=date_string)
179
180     # Check alarm
181     if self.alarm_active and time_string == self.alarm_time:
182         self.set_time_label.after(1000, self.update_time)
183
184     self.set_time_label()
185     try:
186         hour = int(self.time_var.get())
187         minute = int(self.minute_var.get())
188         second = int(self.second_var.get())
189
190         if not (0 < hour < 24 and 0 < minute < 60 and 0 < second < 60):
191             messagebox.showerror("Invalid Time", "Please enter valid time values!")
192
193         self.alarm_time = f"{hour:02d}:{minute:02d}:{second:02d}"
194         self.alarm_active = True
195
196         self.time_label.config(text="Alarm set for: " + self.alarm_time, fg="lime")
197         self.set_button.config(state="disabled")
198         self.cancel_button.config(state="normal")
199
200         messagebox.showinfo("Alarm Set", "Alarm set for " + self.alarm_time)
201
202     except:
203         messagebox.showerror("Invalid Input", "Please enter numeric values!")

def cancel_alarm(self):
    self.alarm_active = False

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PS Python + v - i 3137 D
```

PS C:\Users\abubka\Desktop\Digital Clock\digital_clock> python digital_clock.py

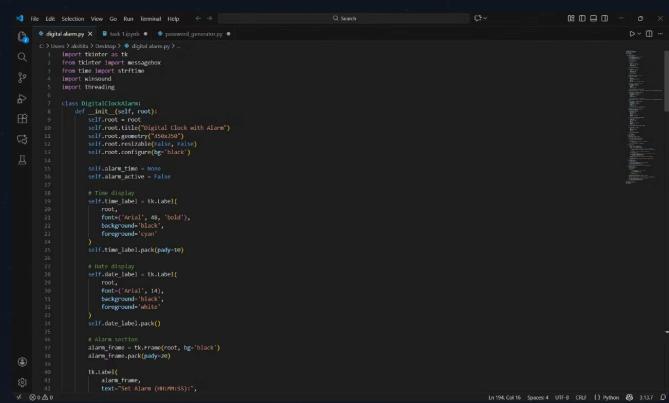


```
File Edit Selection View Go Run Terminal Help ⌘ S Search Q Search C S C

C:\Users\abubka\Desktop\Digital Clock\digital_clock.py
170     self.cancel_alarm(self):
171         self.alarm_active = False
172         self.time_label.config(text="No alarm set", fg="yellow")
173         self.set_button.config(state="normal")
174         self.cancel_button.config(state="disabled")
175
176         messagebox.showinfo("Alarm Cancelling", "Alarm has been cancelled")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PS Python + v - i 3137 D
```

PS C:\Users\abubka\Desktop\Digital Clock\digital_clock> python digital_clock.py



```
File Edit Selection View Go Run Terminal Help ⌘ S Search Q Search C S C

C:\Users\abubka\Desktop\Digital Clock\digital_clock.py
170     def play_sound(self):
171         try:
172             self.play_system_beep_sound()
173             for _ in range(10):
174                 winsound.Beep(1000, 500)
175             except:
176                 print("An error occurred while playing the sound")
177                 messagebox.showerror("Error", "An error occurred while playing the sound")
178                 self.set_button.config(state="normal")
179                 self.cancel_button.config(state="disabled")
180                 self.root.mainloop()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PS Python + v - i 3137 D
```

PS C:\Users\abubka\Desktop\Digital Clock\digital_clock> python digital_clock.py

