

SUPER PROMPT

TaSched — Task Scheduler & Countdown Orchestrator

PRODUCT IDENTITY

Application Name: TaSched

Meaning: *Task Scheduler*

Positioning: A professional, desktop-first task scheduling and countdown orchestration tool for focused work, meetings, academic sessions, and enterprise workflows.

CORE PRODUCT GOAL

TaSched is a **task scheduling and countdown orchestration application** that allows users to:

- Schedule **single or multiple tasks** for up to **24 hours**
- Run tasks **automatically in sequence**
- Show **pre-alert warnings** as countdown approaches
- Display **time-up alerts** at task completion
- Continue running **while minimized**
- Auto-run **batched tasks**
- Show **scrolling ticker messages** while tasks are running
- Support **multiple visual themes**

The setup window must **auto-hide once the run window starts**, and the user must be able to restore or minimize the app freely while work continues.

THEME SYSTEM (MANDATORY)

TaSched **must support three selectable UI themes**, configurable in Settings and applied consistently across all windows.

1 WAEC Theme (Default – Institutional)

Purpose: Official, authoritative, examination and academic environments.

- Background: **WAEC Navy Blue** #002147
- Primary Text: **White** #FFFFFF
- Accent 1: **WAEC Gold** #FFB800
- Accent 2 (Warnings): **Red** #FF4444
- Accent 3 (Clock / Highlights): **Yellow** #FFD700
- Footer / Metadata: Gold + White mix
- Mood: Formal, calm, authoritative

2 Corporate Theme (Neutral / Professional)

Purpose: Enterprise, office productivity, consulting, general business use.

- Background: **Dark Slate / Charcoal** #1E293B
- Primary Text: **White** #FFFFFF
- Accent 1: **Steel Blue** #3B82F6
- Accent 2 (Warnings): **Amber** #F59E0B
- Accent 3 (Success): **Green** #22C55E
- Mood: Clean, modern, professional

3 Indigenous Theme (Cultural / Grounded)

Purpose: African-inspired, heritage-focused, grounded authenticity.

This theme must use GREEN background

- Background: **Deep Green** #0F3D2E

- Primary Text: **White** #FFFFFF
- Accent 1: **Gold / Yellow** #FFD700
- Accent 2 (Warnings): **Earth Red** #B91C1C
- Accent 3: **Soft Cream** #FAF3E0
- Mood: Warm, grounded, culturally rooted

ARCHITECTURE REQUIREMENTS (MUST FOLLOW)

Use a **modular architecture** (even if a single-file prototype is produced initially). Clearly separate concerns.

1 UI Layer

- Setup Window (Schedule Builder)
- Run Window (Countdown Display)
- Warning Popup Window
- Time-Up Window (fullscreen or popup)
- Ticker Overlay (scrolling message)

2 Core Layer

- Task model
- Schedule model
- Scheduler Engine (state machine)
- Warning Engine (threshold evaluation)
- Time Service (clock + formatting utilities)

3 Services Layer

- Audio Service (pygame-based, no playsound)
- Storage Service (JSON persistence)
- Logging Service (run history)
- Resource Service (safe asset loading, PyInstaller-friendly)
- Theme Service (apply theme colours consistently)

FUNCTIONAL REQUIREMENTS (MANDATORY)

A) Schedule Builder (Setup Window)

- Create tasks with:
 - Title
 - Duration (minutes/seconds)
 - Mode: sequential OR absolute (clock-based)
 - Warning thresholds (e.g. 10/5/1 min)
 - Sound profile:
 - Warning sound
 - Time-up sound
 - Display options:
 - Fullscreen or popup time-up
 - Ticker enabled/disabled
 - Ticker text
 - Ticker direction (left/right)
 - Ticker position (top/bottom)
- Add **multiple tasks** into a schedule
- Reorder tasks
- Duplicate tasks
- Remove tasks
- Save schedule as template

- Load templates
- **Start Schedule button must auto-hide setup window**

B) Run Window (Timer Display)

- Fullscreen by default
- Toggle fullscreen (F11)
- Minimize and restore without stopping timer
- Displays:
 - Schedule name
 - Current task name
 - Countdown time
 - Real-time clock
- Automatically advances to next task
- Optionally display next-task preview

C) Warnings and Time-Up

- Warning popups at defined thresholds
- Warning sound optional
- Time-up window:
 - Uses background image if available
 - Uses selected theme colours
 - Plays time-up sound
 - Auto-close after configurable seconds
- After time-up:
 - Auto-start next task if available

D) Ticker Overlay

- Horizontal scrolling message while timer runs
- Position: top or bottom
- Direction: left → right OR right → left
- Speed: slow / medium / fast
- Text source:
 - Custom
 - Auto-generated (Current Task | Next Task)

E) Audio System (MANDATORY)

- Use **pygame.mixer**
- Initialize mixer once
- Provide play_sound(path) that:
 - Stops/unloads previous sound
 - Prevents overlapping/jamming
 - Supports warning vs time-up logic
- App must continue gracefully if audio device fails

F) Assets and Resource Handling

Assets exist in app root:

- WAEC_Background.png
- WAEC_Logo.webp
- WAEC_Tone.mp3
- WAEC_Icon.ico

Implement:

- resource_path()
- find_asset()

Must work in **development and PyInstaller builds**.

G) Persistence

Use local JSON storage:

- settings.json
- templates.json
- schedules.json
- logs.txt (append-only)

H) UX RULES (NON-NEGOTIABLE)

- Setup window auto-hides when run starts
- Timer continues when minimized
- Alerts must still appear when minimized
- User can restore windows at any time

GIT DISCIPLINE (MANDATORY)

You **must use Git properly**.

Repository:

<https://github.com/bofe82frank/TaSched>

Rules:

1. Initialize Git if not already done
2. Commit **after every meaningful feature or fix**
3. Use clear commit messages, e.g.:
 - feat: add scheduler engine
 - fix: prevent overlapping audio
 - ui: implement theme switching
4. **Always push to GitHub** after committing
5. Do NOT accumulate uncommitted changes

PACKAGING PREPARATION (DO NOT BUILD MSI YET)

- Define constants:
 - APP_VERSION
 - COMPANY
 - PRODUCT_NAME
 - DEPARTMENT
- Prepare PyInstaller compatibility:
 - --onefile
 - --noconsole
 - --icon
 - --add-data assets
 - Windows version metadata support
- Include a build_exe.md explaining the process

DELIVERABLES REQUIRED FROM YOU

1. Proposed folder structure (and follow it)
2. Full Python implementation (single-file or modular)

3. JSON schema examples for settings, templates, schedules
4. “How to Run” instructions
5. “How to Package with PyInstaller” instructions
6. Notes on cross-platform expansion (do not implement mobile yet)

CONSTRAINTS

- Python 3.13
- Tkinter UI
- pygame for audio
- No playsound
- No heavy UI frameworks
- Code must be readable, commented, and maintainable

STARTING INSTRUCTION

Start by:

1. Creating the folder structure
2. Implementing the core scheduler engine
3. Adding theme definitions and theme switching
4. Committing and pushing to GitHub immediately