Smartwatch Task Scheduler Simulation with Face Detection and Voice Notification

Objective:

The objective of the project is to simulate the primary functionality of a task-oriented smartwatch. The simulation includes secure face-based user authentication, task management features, and real-time voice notifications for due tasks.

System Overview:

1.Face Detection for Access Control

- The simulation uses face detection to authenticate both team members.
- If a familiar face is detected, access to the task schedule system is granted.
- If a non-matching face is detected, access is denied.

2.Task Management Features

- Add, edit, and delete tasks can be done by users.
- Each task has a description and a schedule time (HH:MM format).
- All tasks are displayed in an easily retrievable and readable screen.

3.Voice Reminder System

- The system reads aloud the task schedule at the set time using text-to-speech.
- A window pops up with the task and a "Stop" button.

- If the stop button is not clicked, the reminder resounds every 10 seconds.

Technical Details:

1. User Interface

- Built using Tkinter for the GUI.
- Main components:

  Listbox to display tasks

  Add, Edit, and Delete buttons

  Pop-up windows for edit and alerts

2. Task Storage

- Tasks are saved in a local JSON file (tasks.json).
- Handled by the task_data.py module via load_tasks() and save_tasks() functions.

3. Voice Notifications

- Installed with the pyttsx3 library for offline text-to-speech conversion.
- The notifier.py program continuously polls the current time every minute.
- When a task is overdue, it announces the task and shows a stop-enabled pop-up.

Modules:

- main.py

Main application logic and user interface.

- notifier.py

  Background scheduler of notifications with voice and pop-up notifications.

- task_data.py

  Data handling module for persistently saving tasks.

Conclusion

This project successfully emulates an appointment-scheduling smartwatch with functionalities like user login by face detection, task scheduling, and voice alerts. It demonstrates rudimentary embedded system principles like real-time response, multitasking, user interface, and data storage.