Speechtimer / Redetimer

1. Zweck der Anwendung

Dieses System dient dazu, die vorgeschlagene Dauer der Redebeiträgen bei Anlässen oder Debatten auf einem Monitor/Ipad/Iphone o.ä. anzuzeigen.
Ein zentrales Admin-Panel steuert Start, Stop, Reset sowie Mitteilungen.
Der „Speechtimer"-Bildschirm dient der Anzeige auf einem externen Monitor, z. B. für Vortragende oder das Publikum.
Das System läuft in einem Netzwerk (kein Internet nötig)
Steuerung und Anzeige erfolgen als HTML-Applikation auf jeglichem Device in einem modernen Browser

2. Die beiden Anwendungen

`speechtimer.html (Anzeige)`

Dies ist die Ausgabe für den Redner- oder Publikumsscreen (z. B. Beamer, externes Display).

- Aktualisiert sich automatisch, sobald über das Admin-Panel gesteuert wird.
- Mehrere Instanzen auf verschiedenen Anzeigen zeigen die gleiche Zeit.
- Aufruf im Browser: `http://<server-ip>:<port>/`

`admin.html (Steuerung)`

Dies ist das Admin-Panel zur Steuerung des Timers. Es erlaubt:

- Auswahl einer Redezeit (z. B. 2, 5, 10 Minuten)
- Manuelle Eingabe in Minuten oder Sekunden
- Start, Stop, Reset
- Hinweis-/Vollbildanzeigen
- Korrektur der Zeit (+/-)
- Aufruf im Browser: `http://<server-ip>:<port>/admin

Steuerung und Anzeige

- Gestartete Timer werden automatisch auf allen Screens synchronisiert.
- Neue Screens übernehmen automatisch den aktuellen Status.
- Aktive Buttons (z. B. letzte Zeitwahl) sind visuell hervorgehoben.

4. Erweiterungen und Anpassungen

- Zeiten: Weitere Zeitbuttons können in `admin.html` ergänzt oder entfernt werden.
- Layout: Farben und Darstellung direkt in HTML/CSS anpassbar.
- Hinweise: Texteingaben erscheinen sofort auf allen verbundenen Displays.

5. Technische Grundlagen

- Python 3 + Flask + Flask-SocketIO
- HTML, CSS, JavaScript
- Kommunikation per WebSockets (Echtzeit)
- Läuft vollständig offline im lokalen Netzwerk (LAN/WLAN)

6. Netzwerkbedingungen

- Alle Geräte müssen im selben LAN/WLAN sein
- Firewall darf Port (z. B. 55055) nicht blockieren
- Keine Internetverbindung nötig

7. Installation & Start (Windows)

- Python installieren: `https://www.python.org/downloads/`
  - Beim Setup: "Add Python to PATH" aktivieren(beide checkboxen)

- Python Erweiterungen installieren (in Windows CMD-Fenster):
  - `pip install flask flask-socketio flask_httpauth eventlet`

8. Projektstruktur

Alle Dateien in einen Ordner kopieren (in diesem Beispiel c:\speechtimer)
Es geht aber auch jeder andere Ordner. Wichtig ist, dass app.py in diesem Ordner gestartet wird
Es können mehrere Instanzen gestartet werden, Eine weitere Instanz muss aber eine andere Port Adresse in app.py haben.

```
C:\speechtimer
│   app.py
│   manual.docx
│
├─────────static
│   │   socket.io.min.js
│   │
│   ├─────────fonts
│   │       Arial Monospaced MT.ttf
│   │       Arial MT Medium.ttf
│   │
│   └─────────images
│           image1.jpg
│           image2.jpg
│           image3.png
│
└─────────templates
          admin.html
          speechtimer.html
```

9. Starten

```
cd C:\speechtimer
python app.py
```

Wichtig ist, dass app.py in diesem Ordner gestartet wird. Es könnten mehrere Instanzen gestartet werden, Eine weitere Instanz muss aber eine andere Port Adresse in app.py haben

10. Zugriff

- Anzeige: `http://localhost:55055/`
- Admin: `http://localhost:55055/admin`
- Im LAN: `http://<ip-des-pc>:55055/`

11. Benutzername & Passwort

Zugang zum Admin Bereich ist passwortgeschützt.
In `app.py` definiert unter:

```
# Admin-Zugangsdaten
users = {
    "admin": "password123"
}
```

12. Port anpassen (optional)

Standard-Port ist `55055`.
Ändern in `app.py`:

```
if __name__ == "__main__":
    socketio.run(app, host="0.0.0.0", port=55055, debug=True)
```

13. Zusammenfassung

- Webserver mit Python + Flask
- Admin steuert Timer, Nachrichten & Anzeige
- Alles läuft synchron & offline im lokalen Netzwerk