Computer Voice Assistant - GUI Konzept
Dokument: GUI Design & Implementierung
Datum: 06. Dezember 2025

Seite: {page}

GUI-Konzept

Grafische Benutzeroberfläche für Voice Assistant

Inhaltsverzeichnis

Übersicht

Eine grafische Benutzeroberfläche (GUI) macht den Voice Assistant benutzerfreundlicher und professioneller.

Ziele

- ✅ **Status-Anzeige** - Zeige aktuellen Zustand (Listening, Processing, Speaking)
- ✅ **Visuelles Feedback** - Animationen & Farben
- ✅ **Transkript-Anzeige** - Zeige erkannte Sprache
- ✅ **Einstellungen** - GUI für config.ini
- ✅ **System-Tray** - Minimiert im Hintergrund

Framework-Vergleich

Tkinter vs. PyQt5

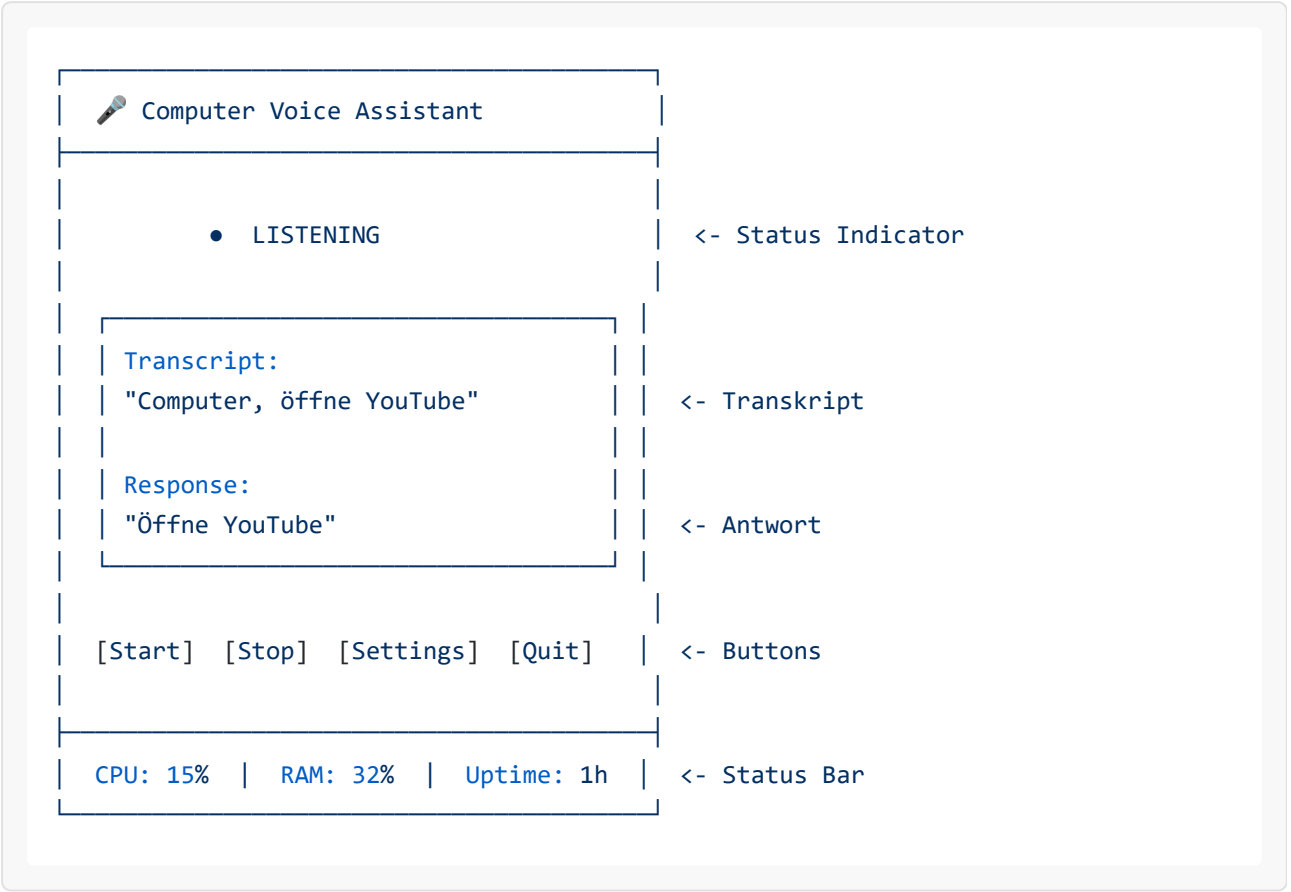
Feature	Tkinter	PyQt5
Einfachheit	⭐⭐⭐⭐⭐	⭐⭐⭐
Design	⭐⭐	⭐⭐⭐⭐⭐
Performance	⭐⭐⭐	⭐⭐⭐⭐
Installation	Built-in	pip install
Lizenz	Python License	GPL/Commercial
Lernkurve	Niedrig	Mittel
Use-Case	Einfache GUIs	Professionelle Apps

Empfehlung

Für Prototyp: Tkinter (schnell, einfach)
Für Produktion: PyQt5 (professionell, schön)

Design-Konzept

Layout



Farb-Schema

Status-Farben:

- 🟢 **Grün** - Listening (bereit)
- 🟡 **Gelb** - Processing (verarbeitet)
- 🔵 **Blau** - Speaking (spricht)
- 🔴 **Rot** - Error (Fehler)
- ⚪ **Grau** - Idle (inaktiv)

Theme:

- **Dark Mode** (Standard) - Schwarz/Dunkelgrau
- **Light Mode** (Optional) - Weiß/Hellgrau

Tkinter-Implementierung

Basis-GUI (Minimal)

```python
#!/usr/bin/env python3
"""
Voice Assistant - Tkinter GUI (Minimal)
"""

import tkinter as tk
from tkinter import ttk
import threading

class VoiceAssistantGUI:
    def __init__(self):
        self.root = tk.Tk()
        self.root.title("🎤 Computer Voice Assistant")
        self.root.geometry("500x400")

        # Status
        self.status = "IDLE"

        # UI erstellen
        self.create_ui()

    def create_ui(self):
        """Erstellt UI-Elemente."""

        # Header
        header = tk.Label(
            self.root,
            text="🎤 Computer Voice Assistant",
            font=("Arial", 18, "bold"),
            bg="#2c3e50",
            fg="white",
            pady=10
        )
        header.pack(fill=tk.X)

        # Status Indicator
        self.status_label = tk.Label(
            self.root,
            text="● IDLE",
            font=("Arial", 24),
```

```
        fg="gray"
    )
    self.status_label.pack(pady=20)

    # Transcript Frame
    transcript_frame = tk.Frame(self.root, bg="white", relief=tk.SUNKEN, bd=2)
    transcript_frame.pack(padx=20, pady=10, fill=tk.BOTH, expand=True)

    # Transcript Text
    self.transcript_text = tk.Text(
        transcript_frame,
        font=("Arial", 12),
        wrap=tk.WORD,
        height=10
    )
    self.transcript_text.pack(fill=tk.BOTH, expand=True, padx=5, pady=5)

    # Buttons
    button_frame = tk.Frame(self.root)
    button_frame.pack(pady=10)

    self.start_btn = tk.Button(
        button_frame,
        text="Start",
        command=self.start_assistant,
        bg="#27ae60",
        fg="white",
        width=10
    )
    self.start_btn.grid(row=0, column=0, padx=5)

    self.stop_btn = tk.Button(
        button_frame,
        text="Stop",
        command=self.stop_assistant,
        bg="#e74c3c",
        fg="white",
        width=10,
        state=tk.DISABLED
    )
    self.stop_btn.grid(row=0, column=1, padx=5)

    tk.Button(
        button_frame,
        text="Settings",
        command=self.open_settings,
```

```
            width=10
        ).grid(row=0, column=2, padx=5)

        tk.Button(
            button_frame,
            text="Quit",
            command=self.root.quit,
            width=10
        ).grid(row=0, column=3, padx=5)

        # Status Bar
        self.status_bar = tk.Label(
            self.root,
            text="Ready",
            bd=1,
            relief=tk.SUNKEN,
            anchor=tk.W
        )
        self.status_bar.pack(side=tk.BOTTOM, fill=tk.X)

    def update_status(self, status: str, color: str):
        """Aktualisiert Status-Anzeige."""
        self.status = status
        self.status_label.config(text=f"● {status}", fg=color)

    def add_transcript(self, text: str, type: str = "user"):
        """Fügt Text zum Transkript hinzu."""
        prefix = "You: " if type == "user" else "Assistant: "
        self.transcript_text.insert(tk.END, f"{prefix}{text}\n\n")
        self.transcript_text.see(tk.END)

    def start_assistant(self):
        """Startet Voice Assistant."""
        self.start_btn.config(state=tk.DISABLED)
        self.stop_btn.config(state=tk.NORMAL)
        self.update_status("LISTENING", "#27ae60")

        # Starte Voice Assistant in Thread
        thread = threading.Thread(target=self.run_assistant, daemon=True)
        thread.start()

    def stop_assistant(self):
        """Stoppt Voice Assistant."""
        self.start_btn.config(state=tk.NORMAL)
        self.stop_btn.config(state=tk.DISABLED)
        self.update_status("IDLE", "gray")
```

```
    def run_assistant(self):
        """Voice Assistant Loop (Dummy)."""
        import time

        while self.status != "IDLE":
            # Simuliere Wake-Word Detection
            time.sleep(2)

            if self.status == "IDLE":
                break

            # Simuliere Erkennung
            self.update_status("PROCESSING", "#f39c12")
            self.add_transcript("Öffne YouTube", "user")
            time.sleep(1)

            # Simuliere Antwort
            self.update_status("SPEAKING", "#3498db")
            self.add_transcript("Öffne YouTube", "assistant")
            time.sleep(1)

            # Zurück zu Listening
            self.update_status("LISTENING", "#27ae60")

    def open_settings(self):
        """Öffnet Settings-Dialog."""
        settings_window = tk.Toplevel(self.root)
        settings_window.title("Settings")
        settings_window.geometry("400x300")

        tk.Label(settings_window, text="Settings", font=("Arial",
16)).pack(pady=10)
        tk.Label(settings_window, text="(Coming Soon)").pack()

    def run(self):
        """Startet GUI."""
        self.root.mainloop()


if __name__ == "__main__":
    app = VoiceAssistantGUI()
    app.run()
```

Erweiterte Features

1. Animierter Status-Indikator

```
class AnimatedIndicator(tk.Canvas):
    """Animierter Kreis-Indikator."""

    def __init__(self, parent, size=100):
        super().__init__(parent, width=size, height=size, bg="white",
highlightthickness=0)
        self.size = size
        self.center = size // 2
        self.radius = size // 3

        self.circle = self.create_oval(
            self.center - self.radius,
            self.center - self.radius,
            self.center + self.radius,
            self.center + self.radius,
            fill="gray",
            outline=""
        )

        self.pulse_active = False

    def set_color(self, color: str):
        """Setzt Farbe."""
        self.itemconfig(self.circle, fill=color)

    def pulse(self):
        """Pulsiert (Animation)."""
        if not self.pulse_active:
            return

        # Größe ändern
        self.radius += 2
        if self.radius > self.size // 2:
            self.radius = self.size // 3

        self.coords(
            self.circle,
            self.center - self.radius,
            self.center - self.radius,
            self.center + self.radius,
            self.center + self.radius
```

```
        )

        self.after(100, self.pulse)

    def start_pulse(self):
        """Startet Pulsieren."""
        self.pulse_active = True
        self.pulse()

    def stop_pulse(self):
        """Stoppt Pulsieren."""
        self.pulse_active = False
```

2. Waveform-Visualisierung

```
import numpy as np

class WaveformVisualizer(tk.Canvas):
    """Echtzeit-Waveform-Anzeige."""

    def __init__(self, parent, width=400, height=100):
        super().__init__(parent, width=width, height=height, bg="black")
        self.width = width
        self.height = height
        self.data = np.zeros(width)

    def update(self, audio_data: np.ndarray):
        """Aktualisiert Waveform."""
        # Resample zu Canvas-Breite
        self.data = np.interp(
            np.linspace(0, len(audio_data), self.width),
            np.arange(len(audio_data)),
            audio_data
        )

        # Zeichne
        self.delete("all")

        for i in range(self.width - 1):
            y1 = int(self.height / 2 - self.data[i] * self.height / 2)
            y2 = int(self.height / 2 - self.data[i+1] * self.height / 2)

            self.create_line(i, y1, i+1, y2, fill="#27ae60", width=2)
```

PyQt5-Implementierung

Installation

```
pip install PyQt5
```

Basis-GUI (Professionell)

```
#!/usr/bin/env python3
"""
Voice Assistant - PyQt5 GUI (Professional)
"""

import sys
from PyQt5.QtWidgets import (QApplication, QMainWindow, QWidget, QVBoxLayout,
                             QHBoxLayout, QLabel, QPushButton, QTextEdit,
                             QStatusBar, QFrame)
from PyQt5.QtCore import Qt, QThread, pyqtSignal
from PyQt5.QtGui import QFont, QPalette, QColor

class VoiceAssistantGUI(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("🎤 Computer Voice Assistant")
        self.setGeometry(100, 100, 600, 500)

        # Dark Theme
        self.set_dark_theme()

        # UI erstellen
        self.create_ui()

    def set_dark_theme(self):
        """Setzt Dark Theme."""
        palette = QPalette()
        palette.setColor(QPalette.Window, QColor(53, 53, 53))
        palette.setColor(QPalette.WindowText, Qt.white)
        palette.setColor(QPalette.Base, QColor(25, 25, 25))
        palette.setColor(QPalette.AlternateBase, QColor(53, 53, 53))
        palette.setColor(QPalette.ToolTipBase, Qt.white)
        palette.setColor(QPalette.ToolTipText, Qt.white)
        palette.setColor(QPalette.Text, Qt.white)
        palette.setColor(QPalette.Button, QColor(53, 53, 53))
        palette.setColor(QPalette.ButtonText, Qt.white)
        palette.setColor(QPalette.BrightText, Qt.red)
        palette.setColor(QPalette.Link, QColor(42, 130, 218))
        palette.setColor(QPalette.Highlight, QColor(42, 130, 218))
        palette.setColor(QPalette.HighlightedText, Qt.black)

        self.setPalette(palette)
```

```
    def create_ui(self):
        """Erstellt UI."""
        central_widget = QWidget()
        self.setCentralWidget(central_widget)

        layout = QVBoxLayout()

        # Header
        header = QLabel("🎤 Computer Voice Assistant")
        header.setFont(QFont("Arial", 20, QFont.Bold))
        header.setAlignment(Qt.AlignCenter)
        layout.addWidget(header)

        # Status Indicator
        self.status_label = QLabel("● IDLE")
        self.status_label.setFont(QFont("Arial", 18))
        self.status_label.setAlignment(Qt.AlignCenter)
        self.status_label.setStyleSheet("color: gray;")
        layout.addWidget(self.status_label)

        # Transcript
        self.transcript = QTextEdit()
        self.transcript.setReadOnly(True)
        self.transcript.setFont(QFont("Consolas", 11))
        layout.addWidget(self.transcript)

        # Buttons
        button_layout = QHBoxLayout()

        self.start_btn = QPushButton("Start")
        self.start_btn.clicked.connect(self.start_assistant)
        self.start_btn.setStyleSheet("background-color: #27ae60; color: white;")
        button_layout.addWidget(self.start_btn)

        self.stop_btn = QPushButton("Stop")
        self.stop_btn.clicked.connect(self.stop_assistant)
        self.stop_btn.setEnabled(False)
        self.stop_btn.setStyleSheet("background-color: #e74c3c; color: white;")
        button_layout.addWidget(self.stop_btn)

        settings_btn = QPushButton("Settings")
        button_layout.addWidget(settings_btn)

        quit_btn = QPushButton("Quit")
        quit_btn.clicked.connect(self.close)
        button_layout.addWidget(quit_btn)
```

```
        layout.addLayout(button_layout)

        central_widget.setLayout(layout)

        # Status Bar
        self.status_bar = QStatusBar()
        self.setStatusBar(self.status_bar)
        self.status_bar.showMessage("Ready")

    def update_status(self, status: str, color: str):
        """Aktualisiert Status."""
        self.status_label.setText(f"● {status}")
        self.status_label.setStyleSheet(f"color: {color};")

    def add_transcript(self, text: str, sender: str = "user"):
        """Fügt Transkript hinzu."""
        prefix = "You: " if sender == "user" else "Assistant: "
        self.transcript.append(f"{prefix}{text}\n")

    def start_assistant(self):
        """Startet Assistant."""
        self.start_btn.setEnabled(False)
        self.stop_btn.setEnabled(True)
        self.update_status("LISTENING", "#27ae60")

    def stop_assistant(self):
        """Stoppt Assistant."""
        self.start_btn.setEnabled(True)
        self.stop_btn.setEnabled(False)
        self.update_status("IDLE", "gray")


if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = VoiceAssistantGUI()
    window.show()
    sys.exit(app.exec_())
```

Features

Must-Have

1. ✅ **Status-Anzeige** (Listening, Processing, Speaking)
2. ✅ **Transkript-Anzeige** (User & Assistant)
3. ✅ **Start/Stop Buttons**
4. ✅ **Settings-Dialog**

Nice-to-Have

1. ⏳ **Waveform-Visualisierung**
2. ⏳ **System-Tray-Integration**
3. ⏳ **Hotkey-Support** (Strg+Shift+Space)
4. ⏳ **Theme-Switcher** (Dark/Light)
5. ⏳ **Statistiken** (CPU, RAM, Requests)

Nächste Schritte

1. ✅ Wähle Framework (Tkinter oder PyQt5)
2. ✅ Implementiere Basis-GUI
3. ✅ Integriere Voice Assistant Logic
4. ✅ Teste & Debugge
5. ✅ Erweitere mit Nice-to-Have Features

Dokumentende
