🌙 Overnight Work Prompts - Voice Assistant Projekt

📋 Projekt-Info (für alle KIs)

GitHub: https://github.com/KoMMb0t/voice_assi
System: Windows 11 Mini PC, Python 3.11
Projektordner: C:\Users\ModBot\ki-sprachsteuerung
Virtual Environment: .venv

🤖 MANUS CHAT 2: "Computer Wake-Word Training"

Super-Prompt für autonomes Arbeiten:

```
Hallo Manus! Ich gehe jetzt schlafen und möchte, dass du über Nacht so viel wie
möglich am "Computer" Wake-Word Training vorbereitest.

**PROJEKT-KONTEXT:**
- GitHub: https://github.com/KoMMb0t/voice_assi
- Voice Assistant mit Wake-Word ("hey jarvis"), STT (Vosk), TTS (Edge TTS)
- System: Windows 11, Python 3.11, Projektordner: C:\Users\ModBot\ki-
sprachsteuerung
- Problem: "hey jarvis" hat Doppelerkennungs-Probleme
- Ziel: Eigenes "Computer" Wake-Word trainieren (wie Star Trek)

***DEINE AUFGABE ÜBER NACHT:***

Bitte arbeite AUTONOM an folgenden Aufgaben und erstelle mir fertige
Dokumente/Code/Anleitungen:

### 1. RECHERCHE & VERGLEICH (30 Min)
Recherchiere und vergleiche alle verfügbaren Wake-Word-Training-Methoden:
- OpenWakeWord (aktuell genutzt)
- Porcupine (Picovoice)
- Snowboy (veraltet?)
- Andere Alternativen

***Erstelle:***
- `wake_word_comparison.md` mit Pro/Contra jeder Methode
- Empfehlung für Windows 11 + Python 3.11
- Kosten-Übersicht (falls kostenpflichtig)

### 2. SCHRITT-FÜR-SCHRITT-ANLEITUNG (1 Std)
Erstelle eine DETAILLIERTE Anleitung für die beste Methode:

***Erstelle:***
- `computer_wakeword_training_guide.md` mit:
  - Voraussetzungen (Software, Hardware)
  - Installation (alle benötigten Pakete)
  - Aufnahme-Prozess (wie viele, wie lange, Qualität)
  - Training-Prozess (Befehle, Parameter)
  - Integration in bestehenden Code
  - Testing & Optimierung
  - Troubleshooting
```

3. AUFNAHME-SKRIPT (30 Min)
Erstelle ein Python-Skript für die Aufnahmen:

Erstelle:
- `record_wake_word.py` mit:
  - Automatische Aufnahme von 100-200 "Computer"-Samples
  - Verschiedene Tonlagen (normal, laut, leise)
  - Countdown zwischen Aufnahmen
  - Automatisches Speichern mit Nummerierung
  - Progress-Anzeige

4. INTEGRATIONS-CODE (1 Std)
Erstelle den Code zur Integration des "Computer" Wake-Words:

Erstelle:
- `voice_assistant_computer.py` (Kopie von voice_assistant_edge.py)
- Ersetze "hey_jarvis" durch "computer"
- Passe alle relevanten Stellen an
- Füge Kommentare hinzu, was geändert wurde

5. GITHUB-DOKUMENTATION (30 Min)
Aktualisiere die GitHub-Dokumentation:

Erstelle:
- `WAKE_WORD_TRAINING.md` für GitHub
- Update für `README.md` (Roadmap: Computer Wake-Word ✅)
- `.gitignore` Einträge für Aufnahme-Dateien

6. TESTING-PLAN (20 Min)
Erstelle einen Test-Plan:

Erstelle:
- `wake_word_testing_checklist.md` mit:
  - Erkennungsrate-Tests
  - Falsch-Positiv-Tests
  - Umgebungs-Tests (leise, laut, Hintergrundgeräusche)
  - Stress-Tests

7. TROUBLESHOOTING-GUIDE (20 Min)
Erstelle einen Troubleshooting-Guide:

Erstelle:
- `wake_word_troubleshooting.md` mit:
  - Häufige Probleme & Lösungen
  - Optimierungs-Tipps

  - Performance-Tuning

WICHTIG:
- Erstelle ALLE Dateien als fertige, ausführbare Dokumente/Code
- Nutze Markdown für Dokumentation
- Nutze Python für Code
- Speichere alles im Projektordner (simuliert)
- Gib mir am Ende eine Zusammenfassung mit allen erstellten Dateien

ZUSÄTZLICHE AUFGABEN (falls Zeit):
- Recherchiere alternative Wake-Word-Modelle (z.B. "Jarvis", "Alexa", etc.)
- Erstelle ein Skript für automatisches Model-Testing
- Plane die LLM-Integration (Architektur-Dokument)

Arbeite autonom und erstelle so viel wie möglich!
Ich schaue morgen früh nach und setze dann alles um.

Danke! 🚀

💬 CHATGPT: Code-Optimierung & Fehleranalyse

Prompt für ChatGPT:

```
Hallo ChatGPT! Ich arbeite an einem Voice Assistant Projekt und brauche deine Hilfe
bei Code-Optimierung.

**Projekt:** https://github.com/KoMMb0t/voice_assi

**Aufgabe 1: Analysiere meinen Code**
Ich habe ein Wake-Word-Doppelerkennungs-Problem. Hier ist mein Code:

[voice_assistant_edge.py Code hier einfügen]

Bitte analysiere:
1. Warum wird das Wake-Word doppelt erkannt?
2. Welche Code-Patterns sind problematisch?
3. Wie kann ich den Audio-Stream besser kontrollieren?

**Aufgabe 2: Optimiere die execute_command() Funktion**
Erstelle eine optimierte Version mit:
- Besserer Struktur (z.B. Command Pattern)
- Einfacherem Hinzufügen neuer Befehle
- Error Handling
- Logging

**Aufgabe 3: LLM-Integration planen**
Erstelle einen Architektur-Plan für:
- Integration von ChatGPT API
- Unterscheidung zwischen "Befehl" und "Frage"
- Fallback-Logik
- Caching für häufige Fragen

Gib mir fertigen, ausführbaren Code!
```

🔍 PERPLEXITY: Recherche & Best Practices

Prompt für Perplexity:

```
Recherchiere bitte folgende Themen für mein Voice Assistant Projekt:

**Thema 1: Wake-Word-Training Best Practices 2025**
- Welche Methoden sind aktuell state-of-the-art?
- Wie viele Aufnahmen braucht man wirklich?
- Welche Tools sind am besten für Windows?
- Gibt es neue Frameworks seit 2024?

**Thema 2: Voice Assistant Architektur**
- Wie strukturieren professionelle Voice Assistants ihren Code?
- Welche Design Patterns werden verwendet?
- Wie integriert man mehrere LLMs effizient?

**Thema 3: Audio-Processing für Wake-Word-Detection**
- Wie verhindert man Doppelerkennungen?
- Welche Audio-Buffer-Strategien gibt es?
- Wie optimiert man die Latenz?

**Thema 4: Open-Source Voice Assistant Projekte**
- Welche erfolgreichen Open-Source Projekte gibt es?
- Was kann ich von ihnen lernen?
- Welche Features sind besonders beliebt?

Gib mir eine Zusammenfassung mit Quellen!
```

🎨 MONICA: Web-Recherche & Asset-Sammlung

Prompt für Monica:

```
Hilf mir bei der Recherche für mein Voice Assistant Projekt:

**Aufgabe 1: Icon-Suche**
Finde kostenlose, kommerzielle Icons für:
- Mikrofon (für Desktop-Icon)
- Voice Assistant (für GitHub)
- Wake-Word (für Dokumentation)

Quellen: icons8.com, flaticon.com, etc.

**Aufgabe 2: Dokumentations-Beispiele**
Finde GitHub-Repos mit exzellenter Dokumentation für Voice Assistants:
- README.md Beispiele
- Wiki-Strukturen
- Diagramme & Visualisierungen

**Aufgabe 3: Tutorial-Videos**
Finde YouTube-Tutorials zu:
- Wake-Word-Training
- OpenWakeWord Usage
- Voice Assistant Development

Gib mir Links und Zusammenfassungen!
```

🤖 CLAUDE/COPILOT: Code-Review & Refactoring

Prompt für Claude/Copilot:

```
Bitte führe ein Code-Review für meinen Voice Assistant durch:

**Code:** https://github.com/KoMMb0t/voice_assi

**Review-Fokus:**
1. Code-Qualität & Best Practices
2. Performance-Optimierungen
3. Error Handling
4. Dokumentation (Docstrings, Kommentare)
5. Sicherheit (API-Keys, etc.)

**Refactoring-Vorschläge:**
- Wie kann ich den Code modularer machen?
- Sollte ich Klassen statt Funktionen nutzen?
- Wie trenne ich besser Concerns (Wake-Word, STT, TTS, Commands)?

**Zusätzlich:**
- Erstelle Type Hints für alle Funktionen
- Schreibe Unit Tests für wichtige Funktionen
- Erstelle eine requirements.txt mit Versions-Pinning

Gib mir fertigen, refactored Code!
```

📊 ZUSAMMENFASSUNG: Overnight Work Plan

Was die KIs über Nacht erledigen:

KI	Aufgabe	Dauer	Output
Manus Chat 2	Wake-Word Training komplett vorbereiten	3-4h	7+ Dokumente & Code-Dateien
ChatGPT	Code-Optimierung & LLM-Integration	1h	Optimierter Code + Architektur
Perplexity	Recherche & Best Practices	30min	Recherche-Dokument mit Quellen
Monica	Asset-Sammlung & Tutorials	30min	Link-Liste & Ressourcen
Claude/Copilot	Code-Review & Refactoring	1h	Refactored Code + Tests

Gesamt: ~6 Stunden autonome Arbeit!

🌅 MORGEN FRÜH: Dein Action Plan

Wenn du aufwachst:

1. ✅ **Checke Manus Chat 2** - Alle Dokumente & Code durchlesen
2. ✅ **Checke ChatGPT** - Optimierten Code reviewen
3. ✅ **Checke Perplexity** - Recherche-Ergebnisse lesen
4. ✅ **Checke Monica** - Assets & Tutorials anschauen
5. ✅ **Checke Claude/Copilot** - Refactoring-Vorschläge prüfen

Dann:

1. 🎤 **Aufnahmen machen** (1h) - 100-200x “Computer” sagen
2. 🤖 **Modell trainieren** (2h) - Mit Manus-Anleitung
3. 💻 **Code integrieren** (30min) - Mit vorbereitetem Code

4. 🧪 **Testen** (30min) - Mit Test-Checklist
5. 🚀 **Auf GitHub pushen** (10min) - Mit neuer Doku

Bis Mittag hast du ein funktionierendes "Computer" Wake-Word! 🎉

📝 Notizen

- Alle KIs arbeiten parallel
- Jede KI fokussiert auf ihre Stärken
- Morgen hast du ~10 fertige Dokumente + optimierten Code
- Du kannst direkt loslegen ohne Planung

Gute Nacht und viel Erfolg! 🌙🚀