# Aarav - Learning AI Personal Assistant

## Complete Setup Guide

### 1. System Requirements

- Python 3.9 or higher
- 8GB RAM minimum (16GB recommended)
- 10GB free disk space
- Microphone and speakers for voice interaction
- GPU recommended for faster AI processing (optional)

### 2. Installation Steps

#### Step 1: Create Virtual Environment

```bash
# Create project directory
mkdir aarav-ai-assistant
cd aarav-ai-assistant

# Create virtual environment
python -m venv aarav_env

# Activate virtual environment
# On Windows:
aarav_env\Scripts\activate
# On Mac/Linux:
source aarav_env/bin/activate
```

#### Step 2: Install Required Packages

Create `requirements.txt`:

```
torch>=2.0.0
transformers>=4.30.0
numpy>=1.24.0
pandas>=2.0.0
scikit-learn>=1.3.0
speechrecognition>=3.10.0
pyttsx3>=2.90
pyaudio>=0.2.11
requests>=2.31.0
sqlite3
nltk>=3.8
spacy>=3.6.0
matplotlib>=3.7.0
seaborn>=0.12.0
plotly>=5.15.0
langchain>=0.1.0
openai>=1.0.0
gradio>=3.50.0
fastapi>=0.100.0
uvicorn>=0.23.0
```

Install packages:

bash

```
pip install -r requirements.txt

# Install PyTorch with GPU support (optional)
pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu121

# Download spaCy model
python -m spacy download en_core_web_sm

# Download NLTK data
python -c "import nltk; nltk.download('vader_lexicon'); nltk.download('punkt')"
```

### Step 3: Additional Setup for Voice (Windows)

bash

```
# If PyAudio installation fails on Windows:
pip install pipwin
pipwin install pyaudio
```

## 3. Enhanced Architecture Components

### A. Learning Modules

#### Personality Learning (`personality_learner.py`):

```python
# Learns user's humor style, formality preferences, response length
# Uses reinforcement learning with user feedback
# Adapts over time based on interaction patterns
```

#### Context Memory (`context_manager.py`):

```python
# Remembers conversation history
# Learns from past interactions
# Maintains long-term and short-term memory
```

#### Preference Engine (`preference_engine.py`):

```python
# Tracks user preferences for different tasks
# Learns optimal times for different activities
# Personalizes automation based on user habits
```

### B. Advanced Features to Implement

#### 1. Emotion Recognition:

```python
# Real-time emotion detection from voice
# Adapts responses based on user's emotional state
# Learns emotional patterns over time
```

#### 2. Habit Learning:

python

```
# Learns user's daily routines
# Proactively suggests actions
# Automates recurring tasks
```

### 3. Communication Style Adaptation:

python

```
# Learns preferred communication channels
# Adapts writing style for emails vs messages
# Learns recipient-specific communication patterns
```

## 4. Training Your Aarav

### Phase 1: Basic Interaction Training (Week 1-2)

python

```
# Daily conversations for 30 minutes
# Provide feedback after each interaction
# Focus on humor and personality preferences
aarav.start_training_mode()
```

### Phase 2: Task Automation Learning (Week 3-4)

python

```
# Teach specific tasks step by step
# Show Aarav how you prefer tasks to be done
# Correct mistakes and provide positive reinforcement
```

### Phase 3: Advanced Personalization (Ongoing)

python

```
# Long-term preference learning
# Seasonal and contextual adaptations
# Multi-modal interaction learning
```

## 5. Advanced Configuration

### Custom Learning Parameters:

python

```python
aarav_config = {
    "learning_rate": 0.001,
    "memory_retention_days": 365,
    "personality_adaptation_speed": "medium",
    "humor_learning_enabled": True,
    "context_window_size": 50,
    "feedback_weight": 0.8,
    "auto_improvement": True
}
```

### API Integration Setup:

python

```python
# OpenAI API for advanced language understanding
OPENAI_API_KEY = "your-api-key-here"

# Google APIs for calendar, email, etc.
GOOGLE_API_CREDENTIALS = "path/to/credentials.json"

# Twilio for SMS
TWILIO_SID = "your-twilio-sid"
TWILIO_AUTH_TOKEN = "your-auth-token"
```

## 6. Advanced Features Roadmap

### Phase 1 Features:

- [x] Basic learning framework
- [x] Personality adaptation
- [x] Memory system
- [ ] Voice recognition
- [ ] Basic task automation

### Phase 2 Features:

- [ ] Email automation
- [ ] Calendar management
- [ ] Research assistance
- [ ] Presentation creation
- [ ] Advanced humor learning

**Phase 3 Features:**

- [ ] Proactive assistance
- [ ] Multi-device synchronization
- [ ] Advanced NLP understanding
- [ ] Custom skill development
- [ ] Integration with smart home devices

## 7. Machine Learning Models Used

1. **Personality Model**: Custom PyTorch neural network
2. **Humor Classification**: Fine-tuned BERT model
3. **Intent Recognition**: Multi-class classification
4. **Emotion Detection**: Sentiment analysis + voice tone analysis
5. **Context Understanding**: Transformer-based sequence modeling

## 8. Data Privacy and Security

- All learning data stored locally
- Encrypted conversation history
- No sensitive data sent to external APIs
- User control over data retention
- Option to reset learning data

## 9. Performance Optimization

**Memory Management:**

```python
# Efficient memory usage for long conversations
# Automatic cleanup of old, less relevant data
# Optimized database queries
```

**Response Speed:**

python

```
# Cached responses for common queries
# Asynchronous processing for complex tasks
# GPU acceleration for AI computations
```

## 10. Troubleshooting Common Issues

### Issue 1: PyAudio Installation Fails

bash

```
# Solution for Windows:
pip install pipwin
pipwin install pyaudio

# Solution for Mac:
brew install portaudio
pip install pyaudio

# Solution for Linux:
sudo apt-get install portaudio19-dev python3-pyaudio
pip install pyaudio
```

### Issue 2: CUDA/GPU Not Detected

bash

```
# Check CUDA installation:
nvcc --version

# Reinstall PyTorch with CUDA:
pip uninstall torch
pip install torch --index-url https://download.pytorch.org/whl/cu121
```

### Issue 3: Voice Recognition Not Working

bash

```
# Test microphone:
python -c "import speech_recognition as sr; r=sr.Recognizer(); m=sr.Microphone(); print('Microp
```

## 11. Getting Started

1. **Run the basic setup:**

```
bash

python aarav_learning_ai.py
```

2. **Start training Aarav:**

- Have conversations daily
- Provide feedback using "feedback: [your comment]"
- Be consistent with your preferences
- Correct mistakes immediately

3. **Expand functionality:**

- Add new skills as separate modules
- Integrate with your existing tools
- Customize the personality further

## 12. Next Steps

After basic setup, you can:

- Add more sophisticated NLP models
- Implement computer vision for visual tasks
- Create a web interface with FastAPI
- Add mobile app connectivity
- Implement multi-user support

Remember: The more you interact with Aarav and provide feedback, the better it becomes at understanding your preferences and humor style!