WavesAI - JARVIS-like AI Assistant for Arch Linux

A powerful, local AI assistant inspired by JARVIS from Iron Man, optimized for Arch Linux with full system control capabilities.

🎯 System Requirements

- **OS**: Arch Linux
- **CPU**: Intel Core i5 12th Gen (or similar)
- **GPU**: NVIDIA RTX 3050 8GB (CUDA support)
- **RAM**: 16GB
- **Storage**: ~10GB free space (for model and dependencies)

✨ Features

Core Capabilities

- 🤖 **Local AI Processing** using Mistral 7B Instruct (Q4_K_M quantized)
- 🖥️ **Full System Control** - CPU, RAM, GPU monitoring and control
- 📦 **Package Management** - Install, remove, update via pacman/yay
- ⚙️ **Process Management** - List, kill, prioritize processes
- 🔧 **Service Control** - Manage systemd services
- 💾 **Storage Management** - Mount/unmount, disk usage, cache cleaning

Advanced Features

- 🌐 **Network Tools** - IP info, ping, speedtest, firewall control
- 🔍 **Intelligent Search** - Files, packages, web search
- 📝 **File Management** - Copy, move, archive, extract
- 💻 **Development Tools** - Git integration, build automation
- 🎨 **Media Control** - Wallpaper, brightness, RGB lighting
- ⏰ **Task Automation** - Cron jobs, scheduled tasks
- 📊 **Real-time Data** - Weather, news, system diagnostics
- 🧠 **Persistent Memory** - Remembers preferences and history

Personality

- Conversational AI responses like JARVIS
- Contextual understanding
- Startup briefings
- Mission mode and diagnostics mode
- Adaptive learning

📥 Installation

Quick Install

```bash
# Download the installation script
wget https://your-repo.com/install_wavesai.sh

# Make it executable
chmod +x install_wavesai.sh

# Run the installer
./install_wavesai.sh
```

Manual Installation

1. Install System Dependencies

```bash
# Essential packages
sudo pacman -S python python-pip base-devel cmake git wget curl \
           nvidia-utils cuda opencl-nvidia brightnessctl openrgb speedtest-cli

# Install yay (AUR helper) if not installed
cd /tmp
git clone https://aur.archlinux.org/yay.git
cd yay
makepkg -si
```

2. Install Python Dependencies

```bash
# Create virtual environment
python -m venv ~/.local/share/wavesai/venv
source ~/.local/share/wavesai/venv/bin/activate

# Install llama-cpp-python with CUDA support
CMAKE_ARGS="-DLLAMA_CUBLAS=on" pip install llama-cpp-python --force-reinstall --no-cache-dir

# Install other dependencies
pip install psutil requests python-dotenv
```

3. Download Mistral Model

```bash
# Create models directory
mkdir -p ~/models

# Download Mistral 7B Q4_K_M (~4.1GB)
wget -O ~/models/mistral-7b-instruct-v0.1.Q4_K_M.gguf \
  https://huggingface.co/TheBloke/Mistral-7B-Instruct-v0.1-GGUF/resolve/main/mistral-7b-instruct-v0.1.Q4_
```

4. Setup WavesAI

```bash
# Create directories
mkdir -p ~/.config/wavesai/{logs,modules}
mkdir -p ~/.local/share/wavesai
mkdir -p ~/.local/bin

# Copy the Python scripts
cp wavesai_core.py ~/.local/share/wavesai/wavesai.py
cp wavesai_modules.py ~/.local/share/wavesai/modules.py
cp wavesai_cli.py ~/.local/bin/wavesctl

# Make them executable
chmod +x ~/.local/share/wavesai/wavesai.py
chmod +x ~/.local/bin/wavesctl

# Create launcher script
cat > ~/.local/bin/wavesai << 'EOF'
#!/bin/bash
source "$HOME/.local/share/wavesai/venv/bin/activate"
python "$HOME/.local/share/wavesai/wavesai.py" "$@"
EOF

chmod +x ~/.local/bin/wavesai

# Add to PATH (add to ~/.bashrc or ~/.zshrc)
echo 'export PATH="$HOME/.local/bin:$PATH"' >> ~/.bashrc
source ~/.bashrc
```

🚀 Usage

Interactive Mode

Start WavesAI in interactive mode:

```bash
wavesai
```

You'll see a startup briefing with system status, then you can interact naturally:

```
[You] ➜ What's my CPU usage?
[WavesAI] ➜ Your current CPU usage is 15.3%, sir. All cores are operating within normal parameters.

[You] ➜ Install vim
[WavesAI] ➜ I can install vim for you. Here's the command:
```bash
sudo pacman -S vim
```
[Execute?] sudo pacman -S vim (y/n): y

[You] ➜ Show me top 5 memory-consuming processes
[WavesAI] ➜ Here are your top memory consumers...
```

CLI Commands

Quick system control without entering interactive mode:

```bash
# System status
wavesctl status

# Show top processes
wavesctl top
wavesctl top --sort memory

# List services
wavesctl services
wavesctl services --state failed

# Network information
wavesctl network

# Search for files
wavesctl search myfile
wavesctl search "*.py" --directory ~/projects

# Package management
wavesctl package search firefox
wavesctl package install firefox
wavesctl package update
wavesctl package clean

# Git operations
wavesctl git status
wavesctl git commit --message "Update files"
wavesctl git push

# Brightness control
wavesctl brightness get
wavesctl brightness set 50

# Kill process
wavesctl kill 1234
wavesctl kill 1234 --force
```

Special Commands

System Information

```
status        - Full system briefing
weather       - Current weather
weather London  - Weather for specific location
search <query>  - Web search via DuckDuckGo
```

Example Conversations

```
[You] ➜ What's the weather like?
[WavesAI] ➜ Currently in Ahmedabad: ⛅ 28°C, Partly Cloudy

[You] ➜ search latest Arch Linux news
[WavesAI] ➜ Searching for: latest Arch Linux news
[Results displayed...]

[You] ➜ Restart network service
[WavesAI] ➜ I can restart NetworkManager for you:
```bash
sudo systemctl restart NetworkManager
```
[Execute?] sudo systemctl restart NetworkManager (y/n):
```

⚙️ Configuration

Edit the configuration file at `~/.config/wavesai/config.json`:

```json
{
  "model_path": "~/models/mistral-7b-instruct-v0.1.Q4_K_M.gguf",
  "context_length": 4096,
  "gpu_layers": 35,
  "threads": 8,
  "temperature": 0.7,
  "max_tokens": 512,
  "personality": "jarvis",
  "enable_notifications": true,
  "auto_execute_safe_commands": false,
  "search_engine": "duckduckgo",
  "default_location": "auto"
}
```

Performance Tuning

For your hardware (RTX 3050 8GB + 16GB RAM):

- **gpu_layers**: 35 (offload 35 layers to GPU)
- **threads**: 8 (utilize all CPU cores)
- **context_length**: 4096 (balance between context and speed)

Adjust based on your needs:

- **More GPU layers** (40-45): Faster but uses more VRAM
- **Fewer GPU layers** (20-30): Slower but saves VRAM
- **Higher temperature** (0.8-1.0): More creative responses
- **Lower temperature** (0.5-0.6): More focused, deterministic

🎮 Advanced Features

Background Daemon Mode

Run WavesAI as a background service:

```bash
# Enable and start the service
systemctl --user enable --now wavesai

# Check status
systemctl --user status wavesai

# View logs
journalctl --user -u wavesai -f
```

Custom Commands

Create custom command scripts in `~/.config/wavesai/modules/`:

```python
# ~/.config/wavesai/modules/custom_commands.py
def my_custom_command():
    """Custom functionality"""
    return "Custom output"
```

Mission Mode

Activate focus mode that monitors resources and minimizes distractions:

```
[You] ➜ activate mission mode
[WavesAI] ➜ Mission mode activated. Disabling notifications and monitoring system performance.
```

Diagnostics Mode

Run comprehensive system health checks:

```
[You] ➜ run diagnostics
[WavesAI] ➜ Running full system diagnostics...
- CPU: ✓ Normal (42°C)
- RAM: ✓ 8.2GB free
- GPU: ✓ Normal (55°C, 2.1GB VRAM used)
- Disk: ⚠ 85% full on /home
- Services: ✓ All critical services running
```

🔒 Security Features

Command Sandbox

WavesAI blocks dangerous commands automatically:

- `rm -rf /`
- `mkfs.*`
- `dd if=`
- Fork bombs and similar

Sudo Integration

WavesAI respects your sudo permissions and will prompt when needed:

```
[You] ➜ Install Docker
[WavesAI] ➜ This requires elevated privileges. The command will use sudo:
```bash
sudo pacman -S docker
```
```

Execution Confirmation

By default, WavesAI asks before executing commands. Set `auto_execute_safe_commands: true` in config to auto-run safe commands.

📊 Examples Use Cases

System Administration

```
[You] ➜ Show me all failed systemd services
[You] ➜ Check disk usage on all partitions
[You] ➜ Find and kill all zombie processes
[You] ➜ Update system and clean package cache
[You] ➜ Show network connections on port 80
```

Development Workflow

```
[You] ➜ Open my project in ~/dev/myapp
[You] ➜ Show git status
[You] ➜ Run tests
[You] ➜ Commit changes with message "Fix bug"
[You] ➜ Push to origin main
```

File Management

```
[You] ➜ Find all PDF files in Documents
[You] ➜ Archive my Downloads folder
[You] ➜ Extract archive.tar.gz to /tmp
[You] ➜ Show largest files in home directory
```

Automation

```
[You] ➜ Schedule system backup at 2 AM every day
[You] ➜ Remind me to take a break in 1 hour
[You] ➜ Clean package cache every week
```

Information Gathering

```
[You] ➜ What's the latest Python version?
[You] ➜ Search for Rust documentation
[You] ➜ Get Arch Linux news
[You] ➜ Check if my public IP changed
```

🐛 Troubleshooting

Model Not Loading

If the model fails to load:

```bash
# Check if model file exists
ls -lh ~/models/mistral-7b-instruct-v0.1.Q4_K_M.gguf

# Verify CUDA installation
nvidia-smi

# Check GPU layers in config
cat ~/.config/wavesai/config.json | grep gpu_layers

# Try with fewer GPU layers
# Edit config.json and set gpu_layers to 20
```

Out of Memory

If you get OOM errors:

```json
// Edit ~/.config/wavesai/config.json
{
  "gpu_layers": 20,       // Reduce from 35
  "context_length": 2048,  // Reduce from 4096
  "threads": 6            // Reduce from 8
}
```

Slow Response Times

```json
// Optimize for speed
{
  "gpu_layers": 40,       // Increase GPU usage
  "max_tokens": 256,      // Reduce max response length
  "temperature": 0.5      // More deterministic
}
```

CUDA Not Detected

```bash
# Reinstall with CUDA support
source ~/.local/share/wavesai/venv/bin/activate
CMAKE_ARGS="-DLLAMA_CUBLAS=on" pip install llama-cpp-python --force-reinstall --upgrade --no-cache-

# Verify CUDA is available
python -c "import llama_cpp; print(llama_cpp.__version__)"
```

Permission Errors

```bash
# Fix ownership
sudo chown -R $USER:$USER ~/.config/wavesai
sudo chown -R $USER:$USER ~/.local/share/wavesai

# Ensure scripts are executable
chmod +x ~/.local/bin/wavesai
chmod +x ~/.local/bin/wavesctl
```

📝 Logs and Debugging

View Logs

```bash
# Main log file
tail -f ~/.config/wavesai/logs/wavesai.log

# System journal (if running as daemon)
journalctl --user -u wavesai -f

# Check last 100 lines
tail -n 100 ~/.config/wavesai/logs/wavesai.log
```

Enable Debug Mode

```bash
# Start with verbose output
WAVESAI_DEBUG=1 wavesai
```

Database Location

Conversation history and preferences are stored in:

```
~/.config/wavesai/memory.db
```

View with:

```bash
sqlite3 ~/.config/wavesai/memory.db "SELECT * FROM memory ORDER BY timestamp DESC LIMIT 10;"
```

🔄 Updates

Update WavesAI

```bash
# Pull latest version
cd ~/wavesai-source
git pull

# Copy updated files
cp wavesai_core.py ~/.local/share/wavesai/wavesai.py
cp wavesai_modules.py ~/.local/share/wavesai/modules.py

# Restart if running as daemon
systemctl --user restart wavesai
```

Update Dependencies

```bash
source ~/.local/share/wavesai/venv/bin/activate
pip install --upgrade llama-cpp-python psutil requests
```

Update Model

Download a different or updated model:

```bash
# Example: Mistral Q5 (higher quality, larger)
wget -O ~/models/mistral-7b-instruct-v0.1.Q5_K_M.gguf \
  https://huggingface.co/TheBloke/Mistral-7B-Instruct-v0.1-GGUF/resolve/main/mistral-7b-instruct-v0.1.Q5_

# Update config.json to point to new model
```

🎨 Customization

Change Personality

Edit the system prompt in `wavesai_core.py` to change personality:

```python
system_prompt = f"""You are WavesAI, a friendly and helpful AI assistant...
# Customize this section for different personalities
```

Add Custom Modules

Create new modules in `~/.config/wavesai/modules/`:

```python
# my_module.py
class MyModule:
    @staticmethod
    def custom_function():
        # Your code here
        pass
```

Import in main script:

```python
from modules.my_module import MyModule
```

Keyboard Shortcuts

Add to your window manager config (Hyprland example):

```
# ~/.config/hypr/hyprland.conf
bind = SUPER, A, exec, kitty -e wavesai
bind = SUPER_SHIFT, A, exec, wavesctl status
```

🌟 Tips and Best Practices

1. **Natural Language**: Talk to WavesAI naturally, like you would to JARVIS
2. **Context Matters**: WavesAI remembers your session, so refer to previous commands
3. **Be Specific**: The more specific your request, the better the response
4. **Review Commands**: Always review commands before executing, especially with sudo
5. **Use CLI for Quick Tasks**: Use `wavesctl` for simple, repetitive tasks
6. **Check Logs**: Regular log review helps catch issues early
7. **Backup Config**: Keep a backup of your customized config.json
8. **GPU Temperature**: Monitor GPU temps during heavy use
9. **RAM Management**: Close WavesAI when running memory-intensive tasks
10. **Keep Updated**: Regularly update dependencies and the model

🤝 Contributing

Contributions are welcome! Areas for improvement:

- Additional system modules
- Better error handling
- Voice input/output support
- GUI dashboard
- Mobile companion app
- Docker/VM management
- More automation features
- Plugin system
- Multi-language support

📄 License

🙏 Acknowledgments

- Mistral AI for the excellent language model
- llama.cpp community for GGUF support
- Arch Linux community
- Iron Man movies for inspiration

📞 Support

- **Issues**: Report bugs on GitHub
- **Documentation**: Full docs at project wiki
- **Community**: Join our Discord/Matrix
- **Updates**: Follow on Twitter/Mastodon

Quick Reference Card

```
╔═══════════════════════════════════════════════════════════╗
║                WavesAI Quick Reference                    ║
╠═══════════════════════════════════════════════════════════╣
║ INTERACTIVE MODE                                          ║
║   wavesai                  Start interactive mode        ║
║   exit / quit / goodbye     Exit WavesAI                  ║
║   status                   System briefing               ║
║   weather [location]        Get weather                   ║
║   search <query>           Web search                    ║
║                                                           ║
║ CLI COMMANDS                                              ║
║   wavesctl status           System status                 ║
║   wavesctl top              Top processes                 ║
║   wavesctl services         List services                 ║
║   wavesctl network          Network info                  ║
║   wavesctl package <action>  Package management           ║
║   wavesctl search <pattern>  File search                  ║
║   wavesctl git <action>     Git operations                ║
║   wavesctl brightness       Control brightness            ║
║   wavesctl kill <pid>       Kill process                  ║
║                                                           ║
║ DAEMON CONTROL                                            ║
║   systemctl --user start wavesai    Start daemon         ║
║   systemctl --user stop wavesai     Stop daemon          ║
║   systemctl --user status wavesai   Check status         ║
║                                                           ║
║ CONFIG & LOGS                                             ║
║   ~/.config/wavesai/config.json     Configuration        ║
║   ~/.config/wavesai/logs/           Log files            ║
║   ~/.config/wavesai/memory.db       Persistent memory    ║
╚═══════════════════════════════════════════════════════════╝
```

WavesAI - Your Personal JARVIS for Arch Linux 🚀