SYSTEM READINESS CHECKLIST

Pre-Implementation Verification for JARVIS Enhancement

Created by: Manus AI (Original Concept Creator)
Date: June 8, 2025
Purpose: Comprehensive system verification before JARVIS implementation

---

🔍 CRITICAL PRE-IMPLEMENTATION CHECKS

IMMEDIATE VERIFICATION REQUIRED

Before beginning JARVIS implementation, verify all existing systems are functional and optimized. This checklist ensures a smooth implementation process without unexpected obstacles.

📋 OLLAMA SYSTEM VERIFICATION

1. Ollama Service Status Check

Windows Command Prompt (Run as Administrator):

```
 # Check if Ollama service is running
tasklist | findstr ollama

# If not running, start Ollama service
ollama serve

# Verify Ollama is responding
ollama list
```

Expected Output:

```
 NAME                    ID              SIZE    MODIFIED
deepseek-r1:latest     [model_id]      [size]  [date]
```

2. DeepSeek R1 Model Verification

Test R1 Model Functionality:

```
 # Test basic R1 model response
ollama run deepseek-r1 "Hello, please confirm you are working
correctly and provide a brief system status."

# Test reasoning capabilities
ollama run deepseek-r1 "Analyze the following: What are the key
considerations for implementing AI on hardware with 4GB VRAM and
16GB RAM?"

# Test code generation
ollama run deepseek-r1 "Write a simple Python function to
monitor GPU memory usage using nvidia-ml-py3."
```

Expected Behavior: - Model loads without errors - Responses are coherent and relevant - No memory or performance issues - Response time under 10 seconds for simple queries

3. Ollama Performance Optimization

Check Current Configuration:

```
 # Check Ollama configuration
ollama show deepseek-r1

# Verify model parameters
ollama show deepseek-r1 --modelfile
```

Optimize for JARVIS Implementation:

```
 # Set optimal parameters for your hardware
# Create custom modelfile for optimized R1

# Example optimized configuration:
# PARAMETER num_ctx 4096
# PARAMETER num_gpu 1
# PARAMETER num_thread 12
# PARAMETER temperature 0.7
```

🖥️ HARDWARE VERIFICATION

1. GPU Status and Driver Check

NVIDIA Driver Verification:

```
 # Check NVIDIA driver status
nvidia-smi

# Verify CUDA installation
nvcc --version

# Check GPU memory and utilization
nvidia-smi -l 1
```

Expected Output:

```
 +-----------------------------------------------------------------------------
+
| NVIDIA-SMI 531.xx       Driver Version: 531.xx       CUDA
Version: 12.x  |
|-------------------------------+----------------------
+----------------------+
| GPU  Name            TCC/WDDM | Bus-Id        Disp.A |
Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-
Util  Compute M. |
|
===============================+======================+==================
|   0  NVIDIA GeForce ... WDDM  | 00000000:01:00.0  On
|                  N/A |
| N/A   45C    P8    15W /  80W |    500MiB /  4096MiB |
2%      Default |
+-------------------------------+----------------------
+----------------------+
```

2. CPU Performance Check

CPU Status Verification:

```
 # Check CPU information
wmic cpu get
name,numberofcores,numberoflogicalprocessors,maxclockspeed

# Monitor CPU usage
wmic cpu get loadpercentage /value
```

```
# Check thermal status (if available)
wmic /namespace:\\root\wmi PATH MSAcpi_ThermalZoneTemperature
get CurrentTemperature
```

3. Memory Status Check

RAM Verification:

```
 # Check total memory
wmic computersystem get TotalPhysicalMemory

# Check available memory
wmic OS get TotalVisibleMemorySize,FreePhysicalMemory

# Monitor memory usage
tasklist /fo csv | findstr /v "Image Name" | for /f "tokens=5
delims=," %i in ('more') do @echo %i
```

🔧 WSL2 ENVIRONMENT VERIFICATION

1. WSL2 Status Check

Windows PowerShell (Run as Administrator):

```
 # Check WSL version and status
wsl --version
wsl --status

# List installed distributions
wsl --list --verbose

# Check if Ubuntu is running
wsl --list --running
```

2. WSL2 GPU Passthrough Verification

Inside WSL2 Ubuntu:

```
 # Check if NVIDIA drivers are accessible
nvidia-smi

# Verify CUDA in WSL2
```

```
nvcc --version

# Test GPU access from WSL2
python3 -c "import torch; print(f'CUDA available:
{torch.cuda.is_available()}'); print(f'GPU count:
{torch.cuda.device_count()}')"
```

3. WSL2 Performance Optimization

Check WSL2 Configuration:

```
# Check WSL2 memory allocation
cat /proc/meminfo | grep MemTotal

# Verify WSL2 configuration file
cat /etc/wsl.conf

# Optimize WSL2 settings if needed
sudo nano /etc/wsl.conf
```

Recommended WSL2 Configuration:

```
[wsl2]
memory=12GB
processors=16
swap=4GB
localhostForwarding=true

[boot]
systemd=true

[interop]
enabled=true
appendWindowsPath=true
```

🐍 PYTHON ENVIRONMENT VERIFICATION

1. Python Installation Check

Both Windows and WSL2:

```
# Check Python version
python --version
python3 --version
```

```
# Check pip version
pip --version
pip3 --version

# Verify virtual environment capability
python -m venv test_env
```

2. AI Framework Installation Check

Critical AI Libraries Verification:

```
# Check PyTorch installation and CUDA support
python -c "import torch; print(f'PyTorch version:
{torch.__version__}'); print(f'CUDA available:
{torch.cuda.is_available()}'); print(f'CUDA version:
{torch.version.cuda}')"

# Check Transformers library
python -c "import transformers; print(f'Transformers version:
{transformers.__version__}')"

# Check other critical libraries
python -c "import numpy, pandas, opencv-cv2 as cv2; print('Core
libraries OK')"

# Check NVIDIA ML Python
python -c "import nvidia_ml_py3 as nvml; nvml.nvmlInit();
print('NVIDIA ML Python OK')"
```

3. Package Updates and Dependencies

Update Critical Packages:

```
# Update pip
python -m pip install --upgrade pip

# Update PyTorch for CUDA 12.x
pip install torch torchvision torchaudio --index-url https://
download.pytorch.org/whl/cu121

# Update Transformers and related packages
pip install --upgrade transformers accelerate bitsandbytes

# Install missing dependencies
pip install nvidia-ml-py3 psutil opencv-python ultralytics
whisper
```

🌐 NETWORK AND CONNECTIVITY CHECK

1. Internet Connectivity Verification

```
# Test basic connectivity
ping google.com -c 4

# Test Hugging Face model hub access
curl -I https://huggingface.co

# Test GitHub access (for repository cloning)
curl -I https://github.com

# Test PyPI access (for package installation)
curl -I https://pypi.org
```

2. Model Download Capability

Test Model Download Speed:

```
# Test download speed for AI models
wget --spider --server-response https://huggingface.co/
microsoft/DialoGPT-medium/resolve/main/pytorch_model.bin

# Check available disk space for models
df -h
```

🔒 SECURITY AND PERMISSIONS CHECK

1. Administrator Privileges

Windows:

```
# Verify administrator access
net session

# Check UAC status
reg query
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System /
v EnableLUA
```

WSL2:

```
# Check sudo access
sudo -v

# Verify user permissions
id
groups
```

2. Firewall and Antivirus Check

Windows Security:

```
# Check Windows Defender status
sc query windefend

# Check firewall status
netsh advfirewall show allprofiles state

# Verify no AI-related processes are blocked
```

📊 PERFORMANCE BASELINE ESTABLISHMENT

1. System Performance Benchmarking

GPU Performance Test:

```
# Create: system_benchmark.py
import torch
import time
import nvidia_ml_py3 as nvml

def gpu_benchmark():
    if not torch.cuda.is_available():
        print("CUDA not available!")
        return

    device = torch.device("cuda")

    # Memory test
    nvml.nvmlInit()
    handle = nvml.nvmlDeviceGetHandleByIndex(0)
    mem_info = nvml.nvmlDeviceGetMemoryInfo(handle)
    print(f"GPU Memory: {mem_info.used/1024**3:.1f}GB used /
{mem_info.total/1024**3:.1f}GB total")
```

```
    # Performance test
    x = torch.randn(1000, 1000, device=device)
    start_time = time.time()
    for _ in range(100):
        y = torch.matmul(x, x)
    torch.cuda.synchronize()
    end_time = time.time()

    print(f"GPU Performance: {(end_time - start_time)*1000:.2f}
ms for 100 matrix multiplications")

if __name__ == "__main__":
    gpu_benchmark()
```

Run Benchmark:

```
python system_benchmark.py
```

2. Ollama Performance Baseline

R1 Model Performance Test:

```
# Test R1 response time
time ollama run deepseek-r1 "What is 2+2? Provide a brief
answer."

# Test R1 with complex reasoning
time ollama run deepseek-r1 "Analyze the pros and cons of using
quantized models for AI inference on consumer hardware."

# Test R1 memory usage during operation
# (Monitor with nvidia-smi in separate terminal)
```

✅ SYSTEM READINESS CHECKLIST

Pre-Implementation Verification:

- [ ] **Ollama Service:** Running and responsive
- [ ] **DeepSeek R1 Model:** Loaded and functional
- [ ] **NVIDIA Drivers:** Latest version installed and working
- [ ] **CUDA:** Properly installed and accessible
- [ ] **WSL2:** Running with GPU passthrough enabled

- [ ] **Python Environment:** All required packages installed
- [ ] **Internet Connectivity:** Stable and fast enough for model downloads
- [ ] **System Permissions:** Administrator/sudo access available
- [ ] **Performance Baseline:** Established and documented
- [ ] **Available Storage:** Sufficient space for additional models (>50GB recommended)

Performance Targets Verified:

- [ ] **GPU Memory:** <500MB baseline usage, 3.5GB available for AI
- [ ] **CPU Usage:** <20% baseline, cores available for AI processing
- [ ] **RAM Usage:** <4GB baseline, 12GB available for AI
- [ ] **R1 Response Time:** <10 seconds for simple queries
- [ ] **Network Speed:** >10Mbps for model downloads

Critical Issues to Resolve Before Implementation:

If any of these fail, resolve before proceeding:

1. **Ollama not responding** → Reinstall Ollama service
2. **R1 model missing/corrupted** → Re-download DeepSeek R1 model
3. **NVIDIA drivers outdated** → Update to latest Game Ready drivers
4. **CUDA not accessible** → Reinstall CUDA toolkit
5. **WSL2 GPU passthrough failing** → Update WSL2 and Windows
6. **Python packages missing** → Install required AI libraries
7. **Insufficient storage** → Free up disk space or add storage
8. **Performance below targets** → Optimize system settings

🚀 POST-VERIFICATION NEXT STEPS

Once All Checks Pass:

1. **Proceed with JARVIS implementation** using `BLACKBOX_AI_IMPLEMENTATION_PROMPT.md`
2. **Use established baselines** for performance monitoring during implementation
3. **Reference this checklist** if issues arise during implementation
4. **Document any system changes** made during JARVIS setup

Ongoing Monitoring:

- **Daily:** Check GPU temperature and memory usage

- **Weekly:** Verify Ollama and R1 model performance
- **Monthly:** Update drivers and AI frameworks
- **As needed:** Re-run this checklist after system changes

📞 TROUBLESHOOTING QUICK REFERENCE

Common Issues and Solutions:

Ollama Service Won't Start:

```
# Kill existing processes
taskkill /f /im ollama.exe
# Restart service
ollama serve
```

R1 Model Loading Slowly:

```
# Check model integrity
ollama show deepseek-r1
# Re-pull if corrupted
ollama pull deepseek-r1
```

GPU Not Accessible:

```
# Restart NVIDIA services
net stop NVDisplay.ContainerLocalSystem
net start NVDisplay.ContainerLocalSystem
```

WSL2 GPU Issues:

```
# Restart WSL2
wsl --shutdown
wsl
```

This comprehensive system check ensures your hardware and software environment is fully prepared for successful JARVIS implementation. Complete all verifications before proceeding with the enhancement project.