Installation Troubleshooting Guide — Meeting Assistant

This guide documents the issues encountered while installing project dependencies on Windows (Python 3.11, CUDA 12.4, PyTorch 2.6.0+cu124) and their resolutions. Each entry includes the symptom, likely cause, step-by-step fix, and verification.

# 1) ERROR: No matching distribution found for ct2-transformers==4.5.0

**Symptom (pip output)**```  
ERROR: Could not find a version that satisfies the requirement ct2-transformers==4.5.0 (from versions: none)  
ERROR: No matching distribution found for ct2-transformers==4.5.0  
```  
**Cause**  
`ct2-transformers` is not a valid PyPI package for this project. It was mistakenly included. `faster-whisper` already brings in the required CTranslate2 runtime.  
  
**Resolution**  
1. Remove the line containing `ct2-transformers==4.5.0` from `backend/requirements.txt`.  
2. Re-install:  
 ```powershell  
 F:\venvs\ Meeting\_AI\Scripts\Activate.ps1  
 cd F:\Projects\Meeting\_Assistant\backend  
 pip install -r requirements.txt  
 ```  
  
**Verification**  
```powershell  
python - << "PY"  
from faster\_whisper import WhisperModel  
print("faster-whisper import OK")  
PY  
```

**Prevention**

* Keep faster-whisper pinned and avoid adding CTranslate2-related extras manually.

# 2) webrtcvad wheel build failure on Windows (MSVC missing)

**Symptom (pip output)**  
```  
Building wheel for webrtcvad (setup.py) ... error  
error: Microsoft Visual C++ 14.0 or greater is required.  
Get it with "Microsoft C++ Build Tools":  
https://visualstudio.microsoft.com/visual-cpp-build-tools/  
```  
  
**Cause**  
`webrtcvad` provides no prebuilt Windows wheels; pip compiles its C extension. The Microsoft C++ Build Tools (MSVC) were not available in the shell session.  
  
**Resolution**  
1. Install Visual Studio Build Tools 2022 with Desktop development with C++ workload (MSVC v143 or newer, Windows 10/11 SDK).  
2. Close all terminals and open a Developer Command Prompt or PowerShell.  
3. Re-run the install.

```

F:\venvs\Meeting\_AI\Scripts\activate

cd F:\Projects\Meeting\_Assistant\backend

python -m pip install --upgrade pip setuptools wheel

pip install -r requirements.txt

```  
  
**Verification**  
```powershell  
python - << "PY"  
import webrtcvad; v = webrtcvad.Vad(2)  
print("webrtcvad OK; mode:", 2)  
PY  
```

**Prevention**

* Perform any package installs or upgrades that require compilation from a **Developer** shell (Native Tools/Developer PowerShell), or load MSVC into your current PowerShell (see Issue 3).

# 3) where cl returns nothing in VS Code terminal

**Symptom**

INFO: Could not find files for the given pattern(s).

**Cause**  
MSVC tools are not on PATH in a normal VS Code terminal. Only Developer shells load them.  
  
**Impact**

1. Day-to-day development and running the app are unaffected.
2. Only building C extensions (e.g., reinstalling webrtcvad) in that terminal would fail.

**Resolution Options**  
**A. Use Developer shell for builds (recommended, simplest)**

* Run all pip install or build steps from:
  + **x64 Native Tools Command Prompt for VS 2022**, or
  + **Developer PowerShell for VS 2022**.

**B. Make VS Code use a Developer shell**

* VS Code → Terminal → **Select Default Profile** → choose **Developer PowerShell for VS 2022**.
* Open a new terminal in VS Code and retry where cl.

**C. Import MSVC env into any PowerShell (one-liner helper)**

1. Add this function to your PowerShell profile (notepad $PROFILE):

```

function Enable-VSBuildTools {

$vswhere = "$Env:ProgramFiles(x86)\Microsoft Visual Studio\Installer\vswhere.exe"

if (-not (Test-Path $vswhere)) { throw "vswhere not found at $vswhere" }

$vsPath = & $vswhere -latest -products \* -requires Microsoft.VisualStudio.Component.VC.Tools.x86.x64 -property installationPath

if (-not $vsPath) { throw "Visual Studio Build Tools not found." }

$vcvars = Join-Path $vsPath "VC\Auxiliary\Build\vcvars64.bat"

if (-not (Test-Path $vcvars)) { throw "vcvars64.bat not found at $vcvars" }

cmd /c "`"$vcvars`" && set" | ForEach-Object {

$pair = $\_ -split '=', 2

if ($pair.Length -eq 2) {

[System.Environment]::SetEnvironmentVariable($pair[0], $pair[1], 'Process')

}

}

Write-Host "VS Build Tools environment loaded for this session."

}

```

1. In a new VS Code terminal:

```

Enable-VSBuildTools

where cl

```  
  
**Verification**

* where cl shows a path under your VS Build Tools installation.
* Reinstalling a C-built package works from that terminal.

**Prevention**

* Keep using the Developer shell for any future pip install that compiles native code.

# 4) Numpy version change during install

**Symptom (pip output excerpt)**

Attempting uninstall: numpy

Found existing installation: numpy 2.1.2

Uninstalling numpy-2.1.2:

Successfully uninstalled numpy-2.1.2

Successfully installed ... numpy-1.26.4 ...

**Cause**

* requirements.txt pins numpy==1.26.4 for broad compatibility. Many scientific/ML libraries still target the 1.x ABI. Pip correctly downgraded your local numpy.

**Resolution**No action required. This is expected.

**Verification**  
```powershell  
python - << "PY"  
import numpy as np; print(np.\_\_version\_\_)  
PY  
# Should print 1.26.4  
```

**Prevention**

* Keep the pin at 1.26.4 unless all dependencies confirm full support for NumPy 2.x.

# 5) General GPU/ASR stack checks

**Purpose**  
Confirm CUDA and ASR stack healthy.  
  
Checks  
```powershell  
python - << "PY"  
import torch  
print(torch.\_\_version\_\_, torch.cuda.is\_available())  
PY  
  
from faster\_whisper import WhisperModel  
model = WhisperModel("base", device="cuda", compute\_type="float16")  
print("faster-whisper model loaded")  
```

**Expected**

* CUDA available: True, device name shows your RTX 3060 Laptop GPU.
* faster-whisper loads a model without error.

# 6) FAISS selection on Windows

**Context**  
Installed faiss-cpu==1.8.0.post1. GPU wheels not available for CUDA 12.4 on Windows.  
  
**Recommendation**

* Start with faiss-cpu for development and early scale.
* If you later need GPU FAISS:
  + Consider a Linux container or WSL2 environment for easier CUDA builds, or
  + Build FAISS from source on Windows with CUDA 12.4 (advanced).

**Verification**  
```powershell  
python - << "PY"  
import faiss, numpy as np  
index = faiss.IndexFlatIP(384)  
xb = np.random.rand(10, 384).astype("float32")  
index.add(xb)  
D, I = index.search(xb[:2], 5)  
print("faiss OK; neighbors shape:", I.shape)  
PY  
```

# 7) Post-install health check for the API skeleton

**Run server**  
```powershell  
cd F:\Projects\Meeting\_Assistant\backend  
.\run\_dev.ps1  
```  
  
**Verify endpoints**  
- http://127.0.0.1:8000/health/ready → {"status": "ok"}  
- http://127.0.0.1:8000/docs loads FastAPI docs.

**Appendix: When to use which terminal**

* **Build or reinstall native packages** (e.g., webrtcvad):  
  Use **x64 Native Tools Command Prompt for VS 2022** or run Enable-VSBuildTools in your VS Code terminal first.
* **Everyday development** (run server, tests, scripts):  
  Any terminal is fine. cl.exe is not required to run the application.