03_Setting Up Project Environment

Chat Conversation

Note: *This is purely the output of the chat conversation and does not contain any raw data, codebase snippets, etc. used to generate the output.*

User Input

so read the mds file and lets gets started tell me give me steps to downlaod dataset first bro and use pythin to see my all system specs too like cpu gpu ssd ram and all and setup venv too

Listed directory PIXEL IMG SEGMENTATION

Listed directory MDS

Listed directory DATASET

Viewed 01 Ignitia Hackathon – Offroad Semantic Scene Segmentation.md

Viewed 02.md

Planner Response

Now let me create the system specs script and set up the venv. Let me also create a task.md to track progress.

Edited relevant file

Edited relevant file

Planner Response

Now let me create the venv and run the system specs script. First, let me set up the virtual environment:

User accepted the command `python -m venv venv`

Checked command status

Checked command status

Planner Response

Venv created. Now let me install psutil in the venv and run the system specs:

User accepted the command `.\venv\Scripts\pip.exe install psutil`

Checked command status

Planner Response

Good, psutil installed. Now let me run the system specs script:

User accepted the command `.\venv\Scripts\python.exe system_specs.py`

Planner Response

The output is a bit jumbled. Let me get the full output cleanly:

Checked command status

Planner Response

Let me pipe the output to a file so I can read it clearly:

User accepted the command `.\venv\Scripts\python.exe system_specs.py 2>&1 | Out-File -Encoding utf8 specs_output.txt`

User accepted the command `.\venv\Scripts\python.exe system_specs.py > specs_output.txt 2>&1`

Planner Response

The emoji characters are likely causing encoding issues on redirect. Let me fix the script to avoid emoji and redirect properly:

User accepted the command `.\venv\Scripts\python.exe -c "import platform, os, sys, shutil; print('=== OS & PYTHON ==='); print(f'OS: {platform.system()} {platform.release()} ({platform.version()})'); print(f'Python: {sys.version}'); print(f'Python Path: {sys.executable}'); print(f'=== CPU ==='); print(f'Processor: {platform.processor()}'); print(f'Architecture: {platform.machine()}'); print(f'Logical Cores: {os.cpu_count()}'); import psutil; freq=psutil.cpu_freq(); print(f'Physical Cores: {psutil.cpu_count(logical=False)}'); print(f'Max Freq: {freq.max:.0f} MHz' if freq else ''); print(f'=== RAM ==='); vm=psutil.virtual_memory(); print(f'Total: {vm.total/(1024**3):.1f} GB'); print(f'Available: {vm.available/(1024**3):.1f} GB'); print(f'Used: {vm.used/(1024**3):.1f} GB ({vm.percent}%)'); print(f'=== DISKS ==='); [print(f'{p.device} [{p.fstype}] Total:{psutil.disk_usage(p.mountpoint).total/(1024**3):.1f}GB Used:`

`{psutil.disk_usage(p.mountpoint).percent}% Free: {psutil.disk_usage(p.mountpoint).free/(1024**3):.1f}GB') for p in psutil.disk_partitions() if p.fstype]"`

Planner Response

The output is being jumbled by the terminal. Let me try a different approach — write a clean no-emoji script and capture the output:

Edited relevant file

User accepted the command `.\venv\Scripts\python.exe check_specs.py`

Planner Response

Now let me read the JSON result:

Viewed [specs_result.json]

Planner Response

Now I have all the system specs. Let me update the task and prepare the final response for the user:

Edited relevant file