**1) Check recent IIS / app errors (optional but useful)**

Run this as-is in **PowerShell (Admin)**:

Get-WinEvent -LogName Application -MaxEvents 100 |

Where-Object { $\_.ProviderName -match 'IIS-W3SVC-WP|IIS Express|Application Error|SideBySide' } |

Select-Object TimeCreated,ProviderName,Id,LevelDisplayName,Message |

Sort-Object TimeCreated |

Select-Object -Last 20

You’ll see the **last 20** app-log entries with time, provider, event id, level, and message. If there’s a crash reason, it’ll show here.

**2) Activate your venv (your cd line had a typo)**

In **PowerShell**, do:

cd C:\inetpub\wwwroot\PNCECS

.\.venv\Scripts\Activate.ps1

python -V

Expected: your prompt gets a (.venv) prefix and python -V prints the version from the venv.

If Activate.ps1 is blocked by policy, run once:  
Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass

**3) Install Waitress (once)**

You had it twice—one is enough:

pip install waitress

Expected: it installs and ends with “Successfully installed waitress …”.

**4) Make sure you have a proper wsgi.py**

In C:\inetpub\wwwroot\PNCECS\wsgi.py, you need an object named **application**. For Flask, for example:

# wsgi.py (at project root)

from app import app # or wherever your Flask app is created

application = app

If your Flask file exposes app = Flask(\_\_name\_\_) in app.py, the above works.

**5) Point PYTHONPATH at your project (session-only)**

$env:PYTHONPATH = "C:\inetpub\wwwroot\PNCECS"

**6) Run the app with Waitress (your listener)**

waitress-serve --listen=127.0.0.1:8000 wsgi:application

Expected console output:

Serving on http://127.0.0.1:8000

Test quickly from the same server:

Invoke-WebRequest http://127.0.0.1:8000/ | Select-Object -ExpandProperty StatusDescription

You should get OK (or whatever your app returns). Keep this window open while testing.

**(Optional but recommended) Put IIS in front as a reverse proxy**

Since Waitress is now listening on **127.0.0.1:8000**, let IIS handle port 80/8080 and forward to Waitress:

1. In **IIS Manager → your site → URL Rewrite → Add Rule(s)… → Reverse Proxy**
   * **Inbound rule**: to http://127.0.0.1:8000/
   * Enable “Rewrite the domain names of the links in HTTP responses”
2. Your minimal web.config (handlers for FastCGI not needed now):

<?xml version="1.0" encoding="utf-8"?>

<configuration>

<system.webServer>

<rewrite>

<rules>

<rule name="ReverseProxyToWaitress" stopProcessing="true">

<match url="(.\*)" />

<action type="Rewrite" url="http://127.0.0.1:8000/{R:1}" />

<serverVariables>

<set name="HTTP\_X\_FORWARDED\_PROTO" value="http" />

<set name="HTTP\_X\_FORWARDED\_FOR" value="{REMOTE\_ADDR}" />

</serverVariables>

</rule>

</rules>

</rewrite>

<httpProtocol>

<customHeaders>

<add name="X-Forwarded-Host" value="{HTTP\_HOST}" />

</customHeaders>

</httpProtocol>

</system.webServer>

</configuration>

Note:  
what is currently working on the server of the MIT is the following webconfig:  
  
<?xml version="1.0" encoding="utf-8"?>

<configuration>

<system.webServer>

<rewrite>

<rules>

<!-- Proxy everything to the backend on 127.0.0.1:8000 -->

<rule name="proxy-all" stopProcessing="true">

<match url="(.\*)" />

<action type="Rewrite" url="http://127.0.0.1:8000/{R:1}" />

</rule>

</rules>

</rewrite>

<directoryBrowse enabled="false" />

</system.webServer>

</configuration>

1. Browse your IIS binding (e.g., http://127.0.0.1:8080/). Requests should flow:  
   **Client → IIS (listener) → Waitress (127.0.0.1:8000) → Your app**.

**Make it run in the background (service) instead of a foreground console**

If you want Waitress to start on boot, use **nssm** (easy) or a Scheduled Task.

**Using nssm (if available):**

nssm install PNCECS-Waitress "C:\inetpub\wwwroot\PNCECS\.venv\Scripts\python.exe" `

"C:\inetpub\wwwroot\PNCECS\.venv\Scripts\waitress-serve.exe" --listen=127.0.0.1:8000 wsgi:application

# Set the working dir to the project root inside NSSM GUI or:

nssm set PNCECS-Waitress AppDirectory C:\inetpub\wwwroot\PNCECS

nssm set PNCECS-Waitress AppEnvironmentExtra "PYTHONPATH=C:\inetpub\wwwroot\PNCECS"

nssm start PNCECS-Waitress

**Using a Scheduled Task (no extra tools):**  
Create C:\inetpub\wwwroot\PNCECS\run\_waitress.ps1:

$env:PYTHONPATH="C:\inetpub\wwwroot\PNCECS"

Start-Process -FilePath "C:\inetpub\wwwroot\PNCECS\.venv\Scripts\waitress-serve.exe" `

-ArgumentList "--listen=127.0.0.1:8000","wsgi:application" `

-WorkingDirectory "C:\inetpub\wwwroot\PNCECS" -WindowStyle Hidden

Register it:

$action = New-ScheduledTaskAction -Execute "powershell.exe" -Argument "-NoProfile -ExecutionPolicy Bypass -File C:\inetpub\wwwroot\PNCECS\run\_waitress.ps1"

$trigger = New-ScheduledTaskTrigger -AtStartup

Register-ScheduledTask -TaskName "PNCECS-Waitress" -Action $action -Trigger $trigger -RunLevel Highest

Start-ScheduledTask -TaskName "PNCECS-Waitress"

**Quick checklist (what “done” looks like)**

* (.venv) active and pip install waitress succeeded
* wsgi.py exposes application
* waitress-serve … wsgi:application prints “Serving on [http://127.0.0.1:8000”](http://127.0.0.1:8000%E2%80%9D/)
* Hitting http://127.0.0.1:8000/ returns your app
* IIS reverse proxy sends http://127.0.0.1:8080/ to Waitress
* Optional: service/task keeps Waitress running after logoff/reboot

If you paste your current wsgi.py (or repo layout), I’ll confirm the exact import line you need for application.