Creating and coding a IPS tracker from scratch optimised for gaming

# Specifications and requirements:

1. Accessing the CPU and GPU temp and usage.
2. Creating an overlay display which won’t impact the visibility, performance and the user experience.
3. Managing windows API and graphical libraries.
4. Real-time management (smooth updates, low consumption).
5. Estimated time: 1-2 weeks for an MVP, around 3-4 weeks for a finished product.

# Technologies:

Python (psutil, GPUtil, py-cpuinfo, wmi.  
C#.  
C++.

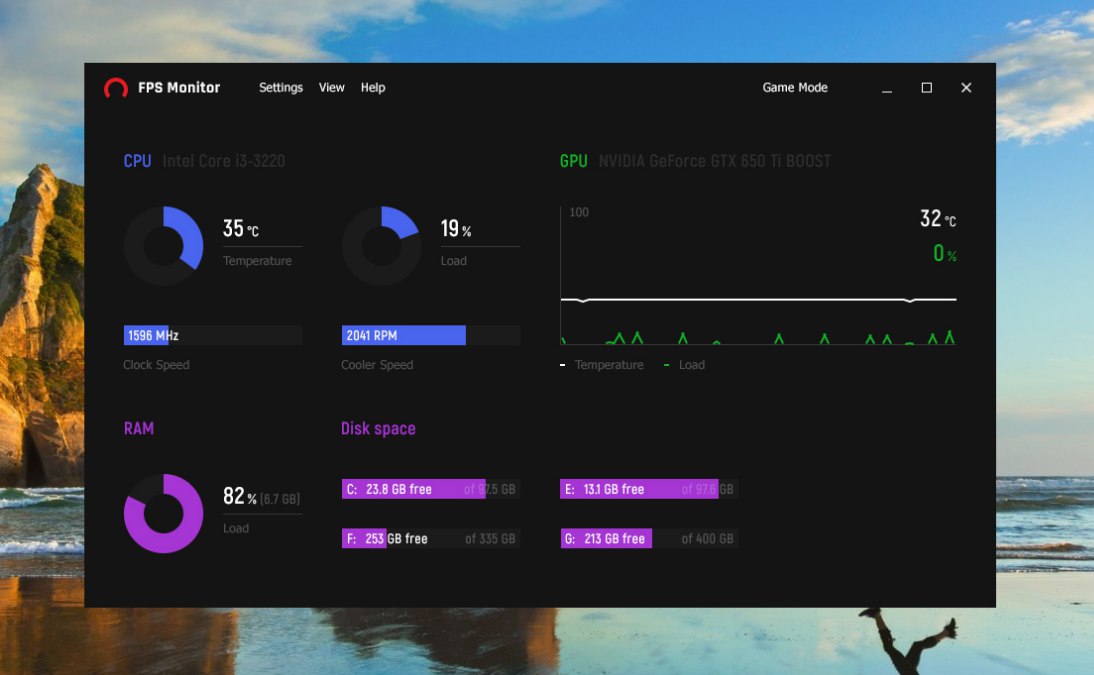
## System overlay:

PyQt5 or pySide6 (transparent window)

## System statistics:

Psutil (CPU usage, RAM and general stats)  
GPUtil (GPU usage and memory)  
Py-cpuinfo or wmi (for CPU temps)  
OpenHardwareMonitor API or pyOHM (CPU/GPU temps)

## Comparing softwares :

<https://www.techpowerup.com/forums/threads/fps-monitor-ingame-hardware-monitoring-overlay.240281/>

Xbox fps tracker:



MSI Afterburner tracker:

## The comparison:

**1. MSI Afterburner with RivaTuner Statistics Server (RTSS)**

**Primary Use:** GPU overclocking, hardware monitoring, and FPS tracking.  
**Key Features:**

* **Comprehensive Metrics:** Tracks FPS, GPU/CPU temps, usage, clock speeds, RAM/VRAM usage, and more 614.
* **Customizable On-Screen Display (OSD):**
  + Adjust font, size, color, and position of metrics 69.
  + Show/hide specific stats (e.g., only FPS or full system details) 14.
* **Advanced Tools:**
  + **OC Scanner:** Auto-overclocks GPUs for stable performance 2.
  + **Fan Control:** Custom fan curves for better cooling 2.
  + **Video Capture:** Records gameplay with minimal performance impact 2.
* **Cross-Brand Compatibility:** Works with all GPUs (Nvidia, AMD, Intel) 2.

**Pros:**  
✅ Highly customizable (e.g., minimalistic FPS counter or full system stats).  
✅ Low performance overhead.  
✅ Integrates with overclocking tools.

**Cons:**  
❌ Requires setup (enabling metrics in Afterburner + RTSS) 6.  
❌ No built-in game optimization (unlike Xbox Game Bar’s performance boosts).

**2. Xbox Game Bar FPS Tracker**

**Primary Use:** Lightweight performance monitoring for Windows gamers.  
**Key Features:**

* **Built into Windows:** No installation needed (Win + G shortcut) 914.
* **Basic Metrics:** FPS, CPU/GPU/RAM usage 914.
* **Pinning:** Overlay can be pinned to stay visible during gameplay 14.
* **Additional Tools:**
  + Screen recording, screenshot capture, and audio controls 14.
  + **Game Optimizer:** Allocates CPU resources to games (requires Norton 360) 2.

**Pros:**  
✅ Instant access (no setup).  
✅ Minimalist design (good for casual users).  
✅ Integrates with other Xbox features (e.g., recording).

**Cons:**  
❌ Limited metrics (no temps, voltages, or detailed hardware stats) 9.  
❌ Less customizable (fixed transparency/color options) 14.  
❌ Overlay disappears if unpinned 14.

## Work plan:

Downaloading needed libraries :  
After downloading Python, setting the env up and testing it, we install our useful libraries by using this code :  
pip install psutil GPUtil py-cpuinfo PyQt5 wmi

After doing this, install and launch OpenHardwareMonitor, open up options tab and search for Web Server and enable it  
open the server locally using :   
→ <http://localhost:8085>

**Authentification** :  
Add a password and configure it for a secure access :

\*Xml\*

<WebServer>

<Enabled>true</Enabled>

<Port>8085</Port>

<Password>VotreMotDePasse</Password>

</WebServer>

**Phase 2 — Base : récupération des données (2-3 jours)**

1. **CPU** :
   * Utiliser psutil.cpu\_percent(interval=1) pour l’utilisation CPU.
   * Utiliser wmi ou OpenHardwareMonitor pour température CPU.
2. **GPU** :
   * Utiliser GPUtil.getGPUs() pour % utilisation et température GPU.
3. **Test** :
   * Faire un script console qui affiche :
   * CPU: 25% | Temp: 50°C
   * GPU: 12% | Temp: 42°C
   * FPS: (placeholder)
4. **FPS tracking** (facultatif pour cette phase) :
   * Lire via API de l’appli cible → difficile.
   * Sinon, mesurer via temps de rafraîchissement d’un affichage test (pas jeu).

**Phase 3 — Interface graphique overlay (3-4 jours)**

1. **Créer fenêtre PyQt5** :
   * Fond transparent (setAttribute(Qt.WA\_TranslucentBackground)).
   * Toujours au-dessus (setWindowFlags(Qt.WindowStaysOnTopHint | Qt.FramelessWindowHint)).
2. **Afficher texte** en temps réel :
   * QLabel mis à jour toutes les 500 ms via QTimer.
3. **Déplacement** :
   * Rendre l’overlay déplaçable par clic souris.
4. **Options** :
   * Fenêtre réglages → cases à cocher :  
     [ ] FPS | [ ] CPU % | [ ] CPU Temp | [ ] GPU % | [ ] GPU Temp
   * Sauvegarder choix en JSON.

**Phase 4 — Optimisation et design (2-3 jours)**

1. Optimiser refresh → pas plus de 1 update / 0.5 sec.
2. Choisir police lisible, couleurs pour chaque stat (vert, jaune, rouge selon seuils).
3. Ajouter possibilité de changer position via drag-n-drop.
4. Sauvegarder paramètres entre sessions.

**Phase 5 — Bonus avancés (optionnel, 1-2 semaines)**

* Lire FPS directement depuis **DirectX/OpenGL overlay** → demande injection DLL (C++/C#).
* Support multi-GPU.
* Graphiques miniatures (courbes utilisation/température).
* Mode réduit pour juste une ligne de stats.

**4. Spécifications finales du logiciel**

* **Type** : Overlay toujours visible, semi-transparent.
* **Plateforme cible** : Windows 10/11.
* **Fonctions** :
  + FPS (optionnel au début)
  + Utilisation CPU (%)
  + Température CPU (°C)
  + Utilisation GPU (%)
  + Température GPU (°C)
  + Cases à cocher pour activer/désactiver chaque affichage.
* **Performance** :
  + Consommation CPU < 2%
  + Rafraîchissement max 2 fois/seconde.
* **UX** :
  + Déplacement libre de l’overlay.
  + Sauvegarde des préférences.
  + Code propre, modulaire.

options a la fin :

1. ajouter le fait de choisir ou place overlay
2. choisir font
3. make a good ui for choosing everything

### Mid way jai remarque un probleme

On ne peux utiliser openhardware pour fps showing donc on va probablement switcher vers msi afterburner apis  
  
re changement de programme, pour msi afterburner, il faut l’installer avec son rtss, ce qui necessite quil soit en arriere plan, par contre une alternative ideale serait presentmon qui se lance aurtomatiquemenrt avec un script et qui depends de directX, donc qui ne consomme pas de rezssources ce qui est optimale pour pas trop perdre de nos ressources

## Plot twist :

A l’aide de la bibliotheque dxcam et deque de collections et la biblio time, jai reussi a faire en sorte que le fps moyen et fps actuel sois affiché, mais il est bloqué a 60hz (62fps) qui est le taux de rafraichissemlent de l’os (Qtimer est bloqué par ca)  
  
alors ce quon peut faire maintenant cest de lier notre code python a une de ces bibliotheques : OpenGL, DirectX, ou le VBlank.

### How to see the cpu/gpu temp? how to attack the problem?

First, the requirements to make it simple :

We don’t wanna install, use a server or a third party monitoring like opemhardware, so what would be the answer? It would be using librehardwaremonitorpy, by using this python library

How to use it? **Download**

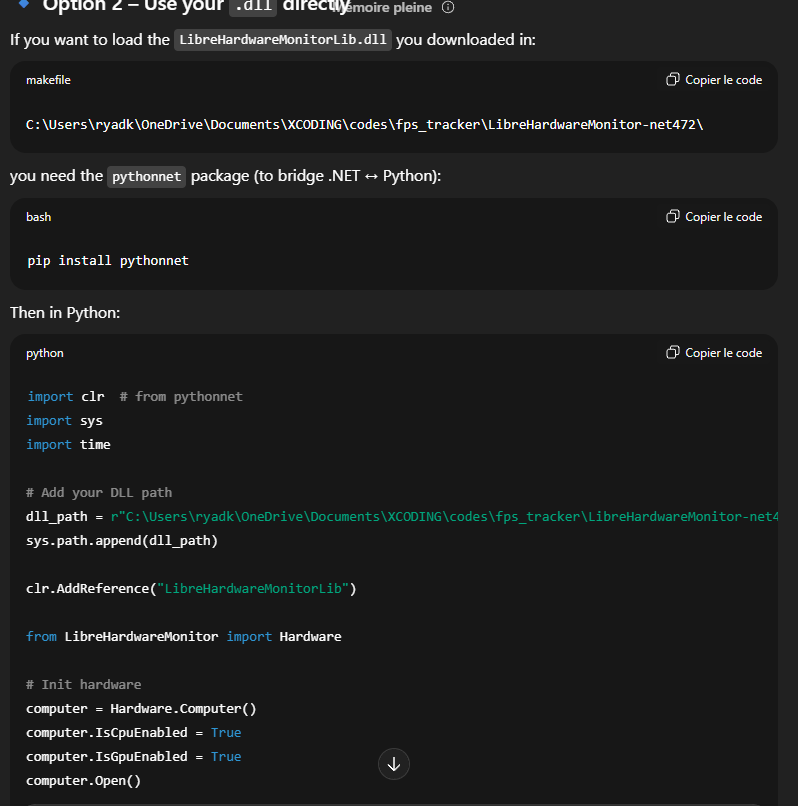
* Go to the official repo: [LibreHardwareMonitor Releases](https://github.com/LibreHardwareMonitor/LibreHardwareMonitor/releases)
* Download the **ZIP** (usually named LibreHardwareMonitor-x.x.x.zip)
* Inside you’ll find LibreHardwareMonitorLib.dll (this is the one Python needs).

Put it in your folder, call it temps\_program or sm but I called it LibreHardwareMonitorLib How to use it?

How to use it??

The link : <https://github.com/LibreHardwareMonitor/LibreHardwareMonitor/releases>

How to implement it?



I had issues with using the LibreHardwareMonitor and importing it, importing and using the .Dll  
finally I found how to fix it and here is how :

