Course Name: Computing with IT Management

Year/Group: 4th Year ITM

Semester: 8

Module Title: IT Management Finals Project

Lecturer Name: Enda Lee, David White, Fernando Perez Tellez

GAME PERFORMANCE & ITMONITORING DASHBOARD

Student Name: Ricardo Danganan Jnr

Student Email: x00191395@mytudublin

Submission Date: 24/03/2025

Due Date: 24/03/2025

Table of Contents

Introduction

The **Game Performance** & **IT Monitoring Dashboard** is my final year IT Management project, developed as part of my 4th-year finals. This real-time monitoring and optimization tool is designed to enhance gaming performance by providing comprehensive system insights.

It tracks CPU and GPU usage, temperatures, RAM consumption, disk activity, VRAM usage, and network performance, ensuring gamers have a clear view of their system's health. Equipped with threshold-based alerts, historical performance tracking, and Steam API integration, it helps optimize system resources, detect installed games, and provide tailored recommendations.

Additionally, I have integrated Grafana monitoring for advanced data visualization and real-time analytics and implemented CSV/JSON data export functionality for deeper analysis. An in-game overlay, powered by Electron, allows players to monitor system performance (FPS, usage, temps) without exiting their game.

The entire dashboard runs as a standalone desktop application via Electron, removing the need to launch it in a browser or rely on localhost, delivering a seamless, app-like experience.

Features

Real-time performance tracking – Monitor CPU, GPU, RAM, VRAM, disk usage, temperatures, and network latency using PowerShell scripts.
System alerts for overheating & resource overuse – Visual and sound alerts using toast notifications when resource thresholds are exceeded.
Standalone desktop app – Powered by Electron, the dashboard runs as a native Windows application without needing a browser or localhost.
In-game overlay (FPS HUD) – Display real-time FPS, CPU/GPU usage, and temperatures inside an always-on-top overlay window.
Historical performance tracking – Logs system data into a local SQLite database with support for CSV/JSON exports by time range.
Grafana integration – Open Grafana dashboard with real-time metrics visualization and historical trends.
Background customization – Switch between animated backgrounds for a personalized dashboard experience.
Steam API integration (Planned) – Detect installed Steam games for future optimization features.
Game optimization (Planned) – Use machine learning to provide tailored performance recommendations (future enhancement).

Usage

View system performance metrics in real-time

- **CPU & GPU Usage & Temperature**: Graphical representation via Chart.js.
- **RAM, VRAM & Disk Usage**: Track memory, VRAM consumption, and storage performance.
- **Network Monitoring**: Display ping, latency, and packet loss for online gaming.
- **Grafana Performance Metrics**: Open Grafana for advanced real-time monitoring and trend analysis.
- **Optimized system monitoring** to reduce overhead, ensuring minimal impact on performance.

Receive alerts when system thresholds are exceeded

- **Automatic notifications** appear when CPU, GPU, RAM, or VRAM usage reaches critical levels.
- **Audio alerts** provide immediate warnings for overheating or resource overuse.
- **Custom alert thresholds** let users adjust limits based on their preferences.

Access the in-game overlay *(Electron FPS HUD)*

- **Live stats inside your game** even when running in **Fullscreen mode**.
- **Displays FPS, CPU/GPU usage, and temperatures** in a compact overlay.
- **Click-through mode enabled**, so the overlay does not interfere with gameplay.
- **Toggle visibility via a hotkey or dashboard button** to show/hide the overlay.

Export historical performance data

- **Download performance logs** as CSV or JSON files based on selected time range (1 hour, 12 hours, 24 hours).
- **Automated background logging** stores real-time data in an SQLite database.
- **Easy access to exported data** for further analysis or sharing.

Customize the dashboard appearance

- **Switch between multiple animated backgrounds** for a personalized UI experience.
- **Adjust UI colours and styles dynamically** for better visibility.

Use the game optimization module *(Upcoming Feature)*

- **Detect installed games** via Steam API.
- **Compare system hardware** to recommended settings.
- **Receive optimization suggestions** based on detected hardware and software.

Dependencies

Frontend:

- **React.js** – For UI components and rendering the dashboard.
- **Chart.js** – For real-time graphical data visualization.
- **React Toastify** – For system alerts and notifications.

- **File-Saver** – For exporting historical data in CSV/JSON format.

Backend:

- **Node.js** – For handling API requests and running the server.
- **Express.js** – For managing API routes and system interactions.
- **WebSockets** – For real-time updates without page refresh.
- **PowerShell Scripts** – For retrieving live system performance data.
- **SQLite** – For storing and retrieving historical system metrics.
- **Child Process (Node.js)** – For executing PowerShell scripts asynchronously.

Electron (Standalone App & In-Game Overlay):

- **Electron.js** – For packaging the dashboard as a desktop app.
- **Electron IPC** – For inter-process communication between the React UI and the FPS overlay.
- **Electron BrowserWindow** – For launching the main dashboard and overlay windows.

APIs & Tools:

- **Steam API (Planned)** – For detecting installed Steam games.
- **NVIDIA API (Planned)** – For AI-driven optimizations and game performance tuning.
- **Grafana** – For external real-time monitoring and visualization.

Core Features of Iteration 1 Successfully Implemented

Live CPU, RAM, Disk, GPU Monitoring ✅

- Tracks real-time **CPU usage, CPU temperature, RAM usage, Disk activity, GPU usage, GPU temperature, VRAM usage, and Network Latency** using PowerShell scripts.

Real-Time Graphs (Chart.js) ✅

- Uses **Chart.js** to display performance metrics as **smooth, interactive line graphs**, updating automatically every few seconds.

System Alerts & Sound Notifications ✅

- Implements **toast alerts and warning sounds** when critical performance limits (high CPU/GPU usage, overheating, network latency) are exceeded.

Historical Data Storage (SQLite) ✅

- Stores real-time performance data in an **SQLite database** (performance_data.db), allowing users to track historical trends over **1hr, 12hr, and 24hr periods**.

Export Data (CSV & JSON) ✅

- Provides an option to **download historical performance logs** in **CSV or JSON format** for further analysis.

Historical Data View ✅

- Allows users to switch between **1hr, 12hr, and 24hr views**, dynamically fetching past performance data from the database.

Core Features of Iteration 2 Successfully Implemented

Separate Electron Window for the Dashboard ✅

- The entire **Game Performance Dashboard now runs inside Electron**, instead of a web browser.
- This allows for **faster performance, better integration with system resources, and improved Fullscreen support**.
- Users can launch the dashboard as a **dedicated app** using npm run start-all.
- This eliminates the need to manually open localhost:5173, making the experience smoother.
- **Optimized Electron Performance** by reducing CPU usage through background throttling, GPU acceleration, and efficient event handling.

Electron Performance Issue & Fixes ✅

During testing, we observed that Electron's CPU usage **fluctuated significantly**, leading to performance concerns when running the overlay. The following optimizations were implemented to reduce CPU consumption:

Issue:

- The Electron overlay would **consume high CPU resources**, especially when running real-time updates.
- requestAnimationFrame() **locked FPS at 60**, causing unnecessary rendering cycles.
- Background processes **kept running at full speed**, even when Electron was minimized.
- 🛠 **Fixes Applied:** ✅ **Lowered data fetching rate** → Reduced polling frequency for CPU/GPU stats to avoid excessive updates.
  ✅ **Optimized FPS counter updates** → Limited requestAnimationFrame() calls to match actual game FPS.
  ✅ **Enabled background throttling** → Prevented Electron from consuming CPU when minimized.
  ✅ **Forced GPU acceleration** → Offloaded rendering to the GPU instead of overloading the CPU.
  ✅ **Improved overlay performance** → Used setIgnoreMouseEvents(true, { forward: true }) to prevent unnecessary event handling.

After these fixes, **Electron now consumes significantly less CPU**, making the dashboard more efficient, even while running in Fullscreen mode.

In-Game Overlay Integration ✅

- Successfully implemented an **Electron-based in-game overlay**, displaying **FPS, CPU usage, GPU usage, CPU temperature, and GPU temperature**.
- The overlay stays **on top of Fullscreen games**, providing real-time performance insights without needing to switch out of the game.

- Features include **adjustable transparency, always-on-top mode, and click-through functionality** to ensure minimal interference with gameplay.
- Future improvements may include **Steam API FPS integration or DirectX hooks for more accurate game FPS tracking**.

Further UI Improvements ☑

- Enhances **visual effects, animation smoothness, and responsiveness** for a **sleek, modern dashboard experience**.

Grafana Optional View (Performance Monitoring Dashboard) ☑

- In addition to the built-in game performance tracking in the Electron dashboard, an optional Grafana-based monitoring view was implemented to provide a more customizable and professional performance dashboard.

What Was Done

- Connected Grafana to SQLite:
- Used an SQLite data source in Grafana to pull system performance data directly from performance_data.db.
- Queries were structured using SQL to fetch time-series data.
- Created Real-Time Performance Panels:
- CPU Usage, GPU Usage, CPU Temp, GPU Temp, VRAM Usage, RAM Usage, Disk Usage, and Network Latency were displayed using gauge and time-series panels.
- Each metric was retrieved dynamically from the SQLite database.
- Added a Button to Open Grafana from Electron:
- A new button was placed below the "Toggle Overlay" button in the Electron dashboard.
- Clicking it opens the Grafana dashboard in a browser, allowing users to view a professional performance monitoring panel.

How to Use

- Start the Grafana Server
- Run grafana-server.exe (or the equivalent startup command in the Grafana installation).
- Open http://localhost:3000/ in a browser to access Grafana.
- Ensure SQLite Database is Available
- The SQLite data source must be correctly set up in Grafana.
- Performance data must be actively logged into performance_data.db.
- Access the Grafana Dashboard from the Electron App
- Click the "View Grafana Dashboard" button to open the optional Grafana view.

Why This Was Added

- Alternative to Built-in Charts → Allows users to leverage Grafana's advanced visualization capabilities.

- Customizable & Expandable → Users can modify and extend the panels without changing the core project code.
- Separate from the Main Dashboard → Provides an alternative method to view system performance, rather than replacing the built-in monitoring.

Planned features for 3rd and Final Iteration

Steam API Integration - Game Detection

- Uses the **Steam API** to fetch a list of installed games, allowing for system performance tracking based on the user's active game library.

Game Optimization Feature

- Compares detected games with system hardware specifications and **recommends performance optimizations** such as adjusting resolution, background processes, and power settings.

Further UI Improvements

- Enhances **visual effects, animation smoothness, and responsiveness** for a **sleek, modern dashboard experience**.

Customizable Performance Alerts and UI Theme

- Future updates will allow users to **adjust threshold levels** for CPU, GPU, RAM, and network alerts, and toggle between **light/dark themes** for the dashboard interface.

Electron App Packaging (Executable Build)

- Convert the Electron-based dashboard into a **standalone executable (.exe) file** for easy distribution and installation.
- This will remove the need for users to run npm commands and simplify deployment on any Windows machine.

Link to Report Recordings (Shared to Enda Lee, David White, Fernando Perez Tellez)

https://tudublin-my.sharepoint.com/personal/x00191395_mytudublin_ie/_layouts/15/stream.aspx?id=%2Fpersonal%2Fx00191395%5Fmytudublin%5Fie%2FDocuments%2FIT%20Management%202024%2D2025%20Files%2F%5B2nd%2DSem%5D%2DProject%2DIterations%2DReports%2FIteration%2D2%2FX00191395%2DProject%2DIteration%2D2%2DReport%2Emp4&referrer=StreamWebApp%2EWeb&referrerScenario=AddressBarCopied%2Eview%2Ed5bd249b%2Db2d4%2D4d11%2D9e75%2Dbb2c067aa078

Link to my GitHub Repository for my Project

https://github.com/ricardodanganan/GamePerformanceMonitoring-Dashboard