**INTERNSHIP REPORT**

**Intel-Unnati**

**On**

**By**

**Y DEEKSHITHA - BU21CSEN0300315**

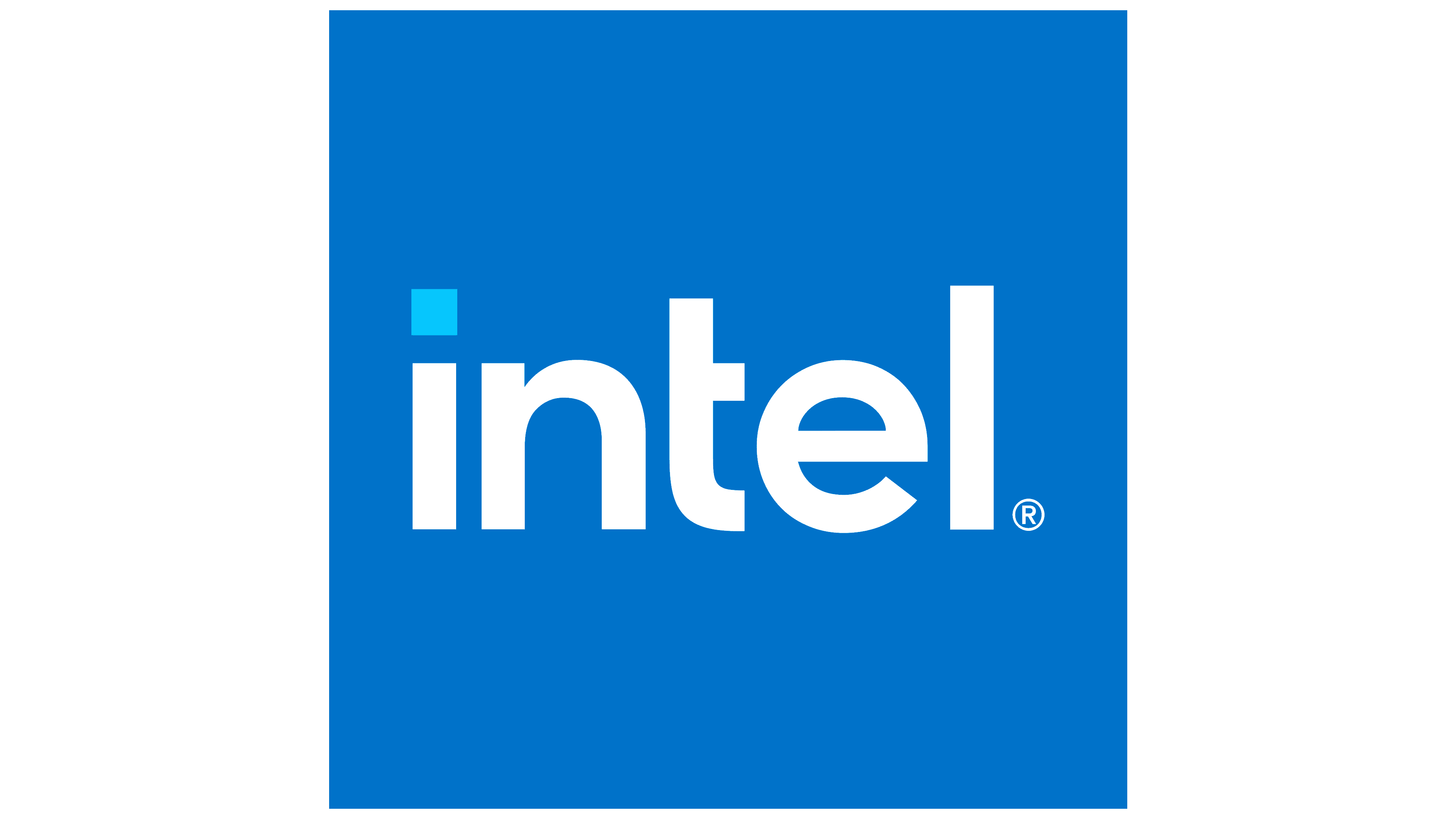
**V MAHESH BABU - BU21CSEN0500123**

**VIDYASHREE K J – BU21CSEN0300454**

**B MOHITH - BU21CSEN0500222**

**T JAYAVARDHAN REDDY - BU21CSEN0500290**

**(Duration: 14-05-2024 to 15-07-2024)**

****

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**Gandhi Institute of Technology and Management**

**(DEEMED TO BE A UNIVERSITY)**

**BENGALURU, KARNATAKA, INDIA**

**SESSION: 2021-2025**

**Abstract:**

In today's rapidly advancing technological landscape, the deployment of 5G networks and edge computing devices has led to a significant increase in power consumption across various locations. This surge in energy usage poses a critical challenge, prompting governmental bodies to advocate for reduced power consumption and the achievement of net-zero power usage. Concurrently, rising electricity costs highlight the importance of accurately understanding and managing the total power drawn by systems to optimize both efficiency and expenses.

**CONTENTS**

1. Introduction --------------------------------------------------------------------------------------------------- 3
2. Problem Statement ------------------------------------------------------------------------------------------- 3
3. Scope ----------------------------------------------------------------------------------------------------------- 3
4. Infrastructure and Requirements --------------------------------------------------------------------------- 3
5. Features -------------------------------------------------------------------------------------------------------- 4-5
6. Process Flow -------------------------------------------------------------------------------------------------- 5
7. Architecture Diagram----------------------------------------------------------------------------------------- 5
8. Justification for Tools and Technologies Used ----------------------------------------------------------- 6
9. Open-Source Tools-------------------------------------------------------------------------------------------- 7
10. System Knobs for Power Measurement ------------------------------------------------------------------- 8
11. Learning Outcomes ------------------------------------------------------------------------------------------- 8
12. Result ----------------------------------------------------------------------------------------------------------- 9
13. Conclusion ---------------------------------------------------------------------------------------------------- 9-10
14. References ----------------------------------------------------------------------------------------------------- 10

**Introduction**

This project's goal is to provide a Power Manager Telemetry application that optimizes power consumption across various systems. The application will collect and analyze real-time telemetry data from hardware sensors (CPU, GPU, memory, peripherals) to monitor power usage, detect inefficiencies, and predict future consumption patterns. It will provide actionable recommendations for power optimization, generate detailed reports and visualizations, and ensure secure data handling through encryption and secure protocols. The application guarantees accurate and reliable power management while maintaining data privacy and security.

**Problem Statement**

Power Manager Telemetry

**Scope**

The scope of this project comprises the creation of a Power Manager Telemetry application that monitors power consumption. The project requires knowledge of hardware telemetry data collection, power management techniques, and Python programming.

**Infrastructure Requirements**

**Hardware:**

* Any x86-based Desktop or Server with Windows

**Software:**

* VS Code
* Python
* Docker Desktop (Windows)

**Requirements:**

* psutil
* CSV
* JSON

**Features**

1. Comprehensive Telemetry Data Collection:

- CPU Utilization Monitoring: Collect real-time CPU usage statistics using the psutil library.

- Memory Utilization Monitoring: Track memory usage, including percentage used and available memory.

- NIC (Network Interface Card) Statistics: Monitor network traffic, including bytes sent and received for each network interface.

- TDP (Thermal Design Power) Placeholder: Placeholder for future integration of TDP data collection to track power consumption related to thermal output.

2. Dynamic Load Simulation:

- Customizable Load Generation: Use Docker containers to simulate different levels of CPU load, adjustable via input parameters.

- Load Simulation Script: Python script (load\_simulator.py) to generate specified CPU utilization, allowing for testing under various load conditions.

3. Automated Data Logging:

- Multi-threaded Logging: Use threading to ensure continuous and efficient data logging without interrupting other processes. ( CSV,JSON files)

4. Robust Data Analysis:

- Average Utilization Calculation: Calculate average CPU, Memory, and NIC utilization over the logged period.

- Dynamic NIC Analysis: Automatically detect all network interfaces, ensuring comprehensive network statistics.

5. Easy Integration and Customization:

- Modular Code Structure: Well-organized codebase that can be easily extended or modified to integrate additional telemetry sources or analysis features.

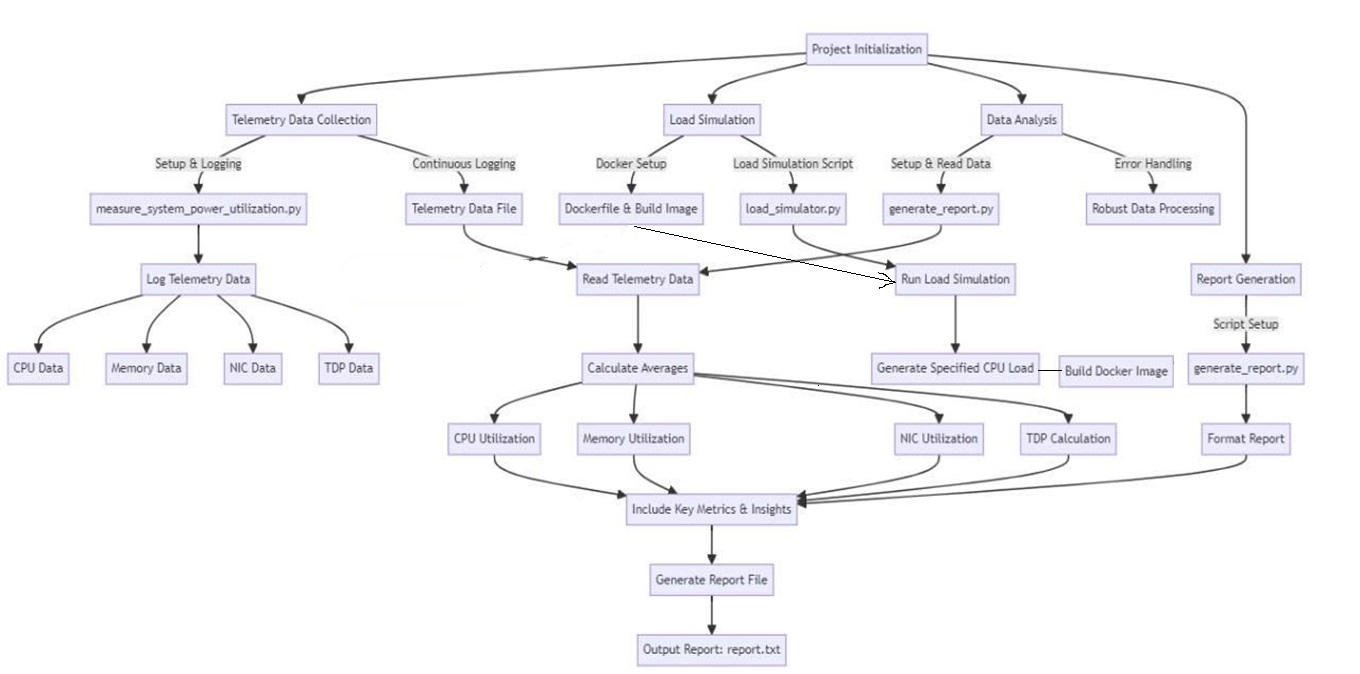
- Docker Integration: Simplified load simulation through Docker, enabling easy deployment and testing across different environments.

7. User-Friendly Operation:

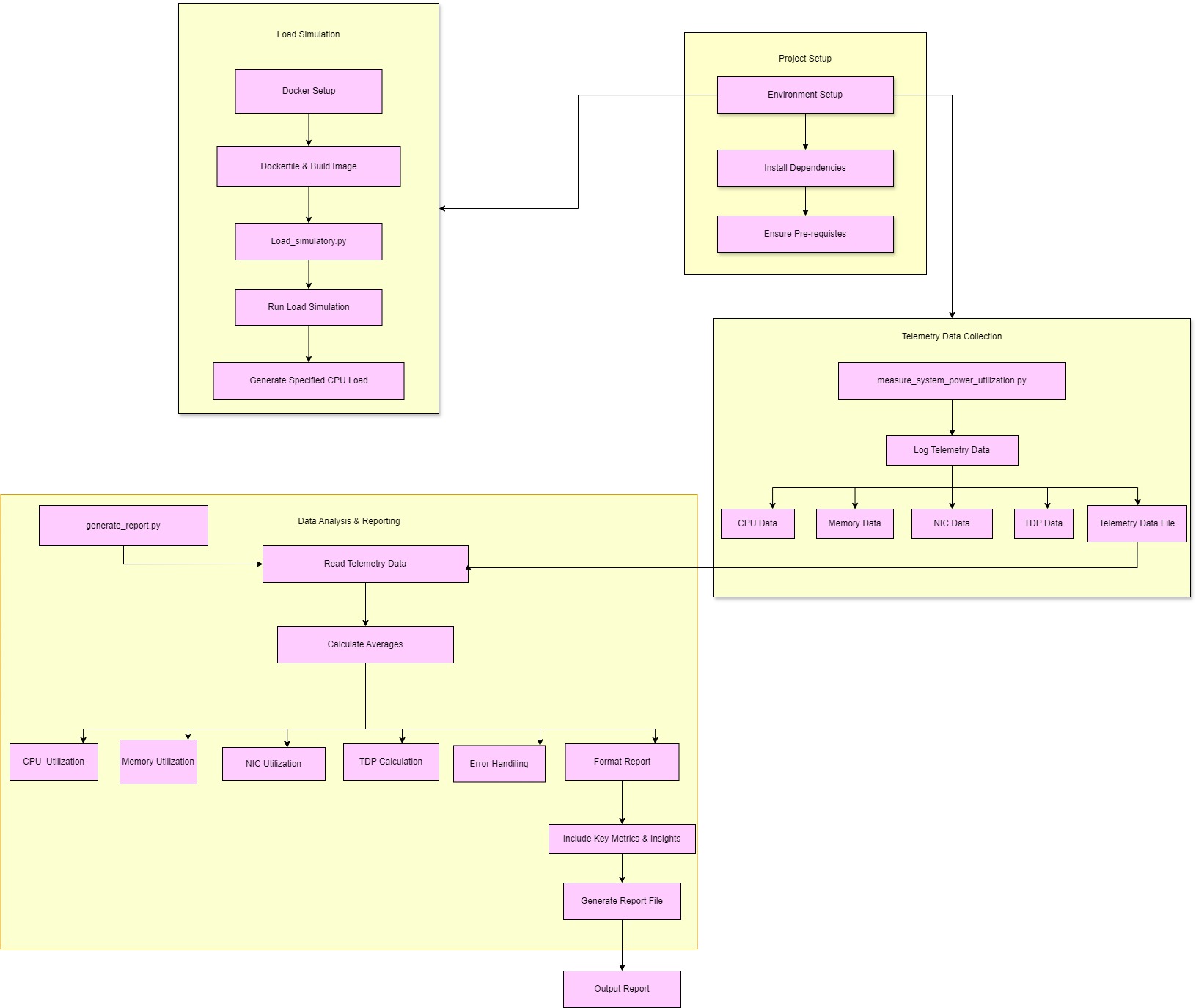
- Simple Command-Line Interface: Run scripts directly from the command line with clear instructions and error messages.

- Automated Processes: Minimize manual intervention with automated data collection, logging, and reporting.

**Process Flow**



**Architecture Diagram**



**Justification for Tools and Technologies Used**

1. Programming Languages:

- Python:- Used for scripting the telemetry data collection, load simulation, and report generation.

- Libraries:

- psutil: For collecting system telemetry data (CPU, memory, NIC).

- JSON: For reading and writing telemetry data.

- threading: For running parallel tasks (data logging).

- subprocess: For running Docker commands from within Python scripts.

2. Containerization:-

Docker:

- Used to create and manage containers for load simulation.

3. Data:

- JSON, CSV:

- Format used for logging telemetry data, enabling easy reading and writing of structured data.

4. Operating Systems:

- Windows:

- Supported environment for developing and running the project with necessary adjustments (e.g., using Docker Desktop).

6. Documentation:

- Markdown:

- Used for writing README files and other documentation.

**Open-Source tools:**

To measure power consumption and gather telemetry data from a system, it is essential to identify and document the various system parameters, often referred to as "knobs," that can be monitored or adjusted. These knobs provide valuable insights into the system's performance and power utilization

**1.PowerShell:** PowerShell is a task automation framework consisting of a command-line shell and scripting language. It is built on .NET and provides powerful tools for managing and monitoring Windows systems.

**Usage:** Use PowerShell scripts to collect telemetry data and measure power utilization. Scripts can be created to fetch data from various system components using Windows Management Instrumentation (WMI) and other system APIs.

**2. Open Hardware Monitor:** Open Hardware Monitor is a free open-source application that monitors temperature sensors, fan speeds, voltages, load, and clock speeds of a computer.

**Usage:** Use this tool to monitor hardware sensors and collect data on CPU, memory, and other components. It can provide real-time telemetry data and log it for analysis.

**4. Intel Power Gadget**: Intel Power Gadget is a software-based power usage monitoring tool enabled for 2nd Generation Intel Core processors or later.

**Usage:** Use this tool to monitor and log the power consumption of Intel processors. It provides detailed information on CPU power usage and can be integrated into scripts for automated data collection.

**5. Perfmon (Performance Monitor):** Perfmon is a Windows tool that provides a visual display of built-in Windows performance counters, either in real-time or as a way to review historic data.

**Usage:** Use Perfmon to monitor various system metrics, including CPU, memory, and network usage. It can be configured to log performance data over time for analysis.

**6. Windows Management Instrumentation (WMI):** WMI is a set of specifications from Microsoft for consolidating the management of devices and applications in a network.

**Usage:** Use WMI queries in PowerShell or other scripting languages to collect detailed telemetry data from Windows systems. It can access a wide range of system information and performance metrics.

**System Knobs for Power Measurement**

To measure power consumption and gather telemetry data from a Windows system, it is essential to identify and document the various system parameters, often referred to as "knobs," that can be monitored or adjusted. These knobs provide valuable insights into the system's performance and power utilization.

**1. CPU Utilization**

* Knob: CPU Load Percentage
* Description: Measures the current load on the CPU as a percentage.

**2. Memory Utilization**

* Knob: Memory Usage Percentage
* Description: Indicates the percentage of total physical memory currently in use.

**3. Network Utilization (NIC)**

* Knob: Network Adapter Bytes Sent/Received per Second
* Description: Measures the amount of data sent and received by the network adapter per second.

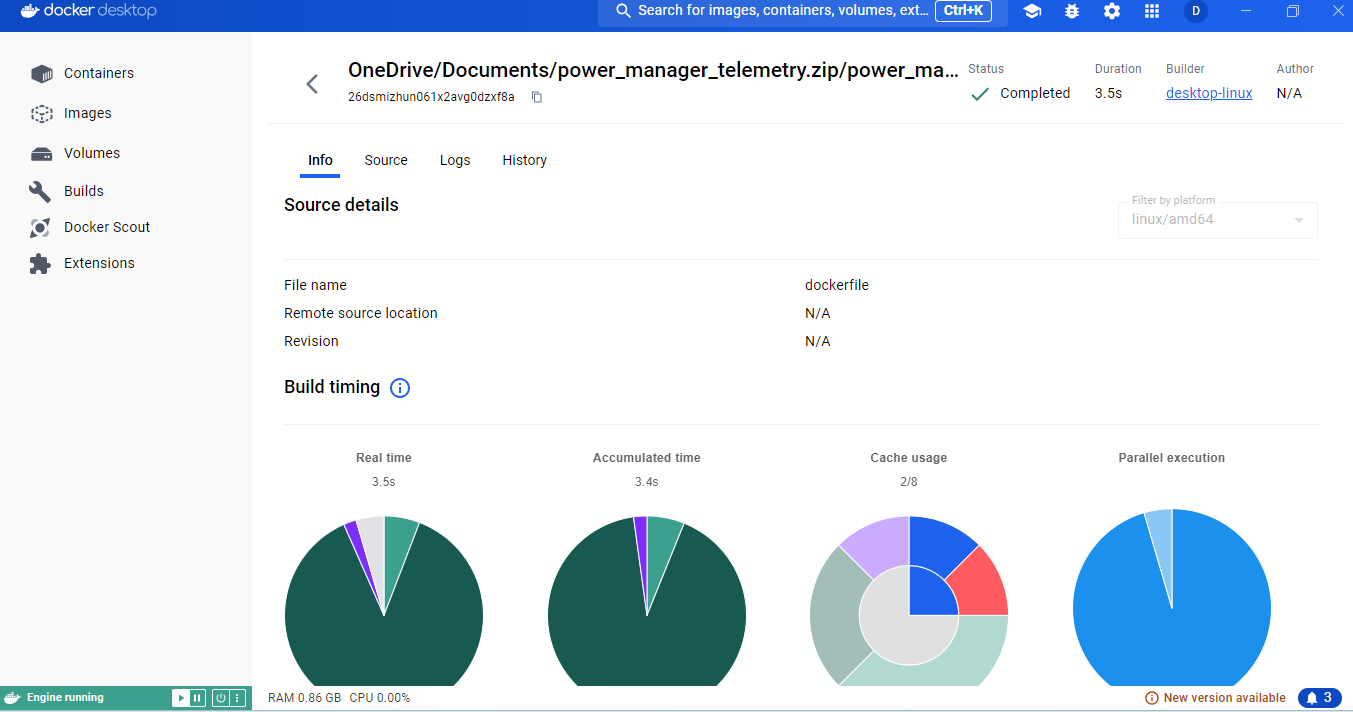
**4. \*Thermal Design Power (TDP)**

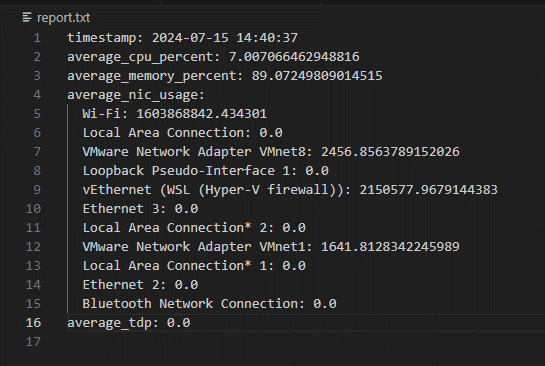
* Knob: CPU Power Consumption (TDP)
* Description: Indicates the power consumption of the CPU. This is usually a fixed value but can be monitored for changes.

**Learning Outcomes**

1. **Data Collection:** Learn to collect real-time telemetry data from hardware components (CPU, GPU, memory, peripherals).
2. **Python Skills:** Improve Python programming, especially in data collection, analysis, and visualization.
3. **Docker:** Gain experience with Docker for creating consistent development and deployment environments.
4. **Testing:** Use load simulators to test the application's performance under various conditions.
5. **Project Management:** Develop skills in managing and executing a project from start to finish.

**Result**





### Conclusion

### The Power Manager Telemetry project represents a significant step forward in addressing the challenges of power consumption in modern computing environments, particularly in the context of 5G and edge computing deployments. By leveraging a combination of Python scripting, and Docker containerization this project provides a comprehensive monitoring solution.

### In conclusion, the Power Manager Telemetry project not only addresses a critical issue in modern computing but also provides a scalable and adaptable framework for ongoing research and development. Through detailed data collection and insightful reporting, this project contributes to the broader goal of sustainable and efficient computing practices.

### References

<https://docs.amd.com/r/en-US/ug907-vivado-power-analysis-optimization/Power-Measurement-Techniques>

<https://docs.python.org/3/>

<https://docs.docker.com/>

<https://sematext.com/glossary/telemetry/#:~:text=Telemetry%20is%20the%20practice%20of,and%20behavior%20of%20a%20sy>