**INTERNSHIP REPORT**

**Intel-Unnati**

**On**

**By**

**Y DEEKSHITHA - BU21CSEN0300315**

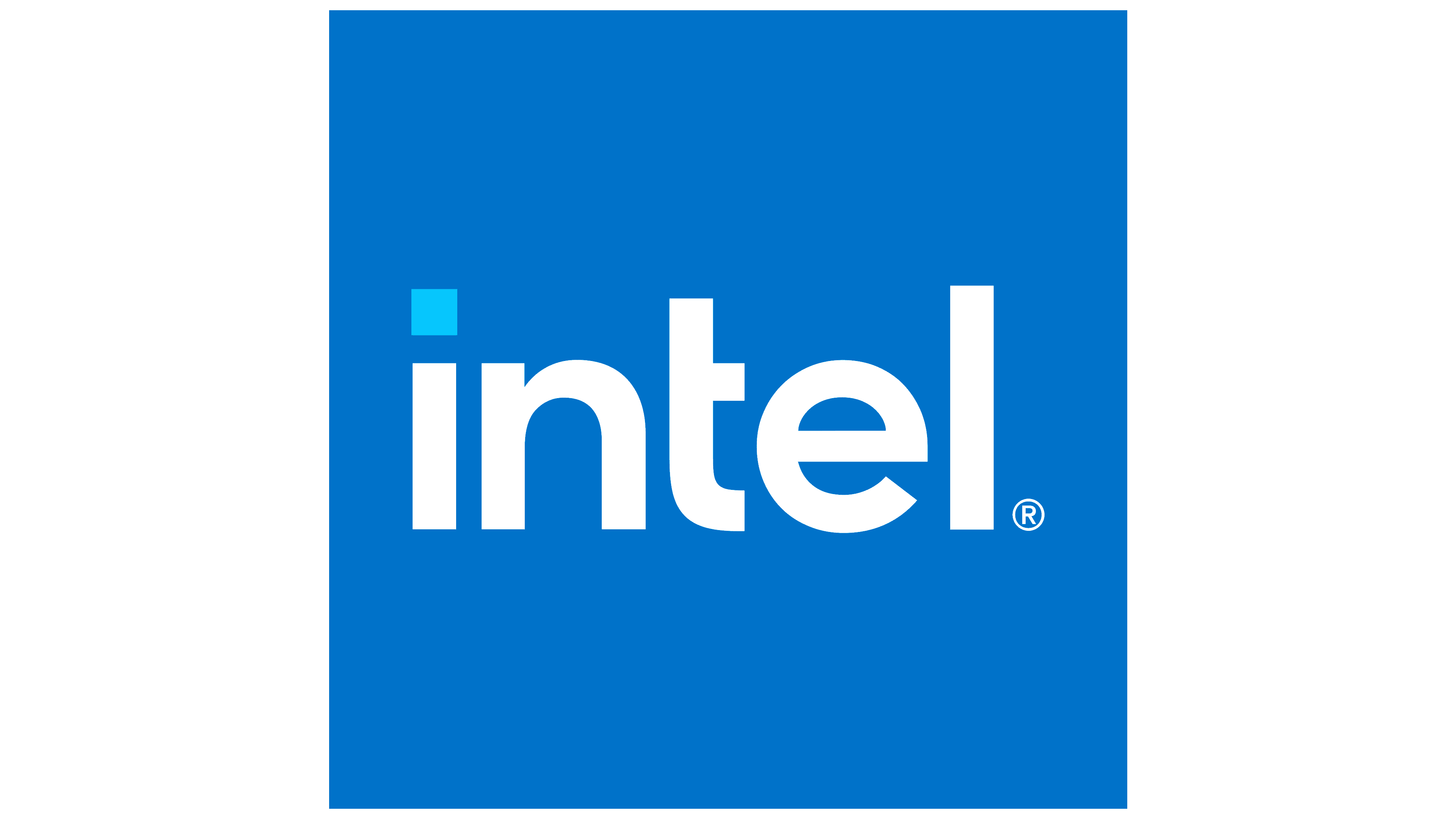
**V MAHESH BABU - BU21CSEN0500123**

**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:**

This project presents a comprehensive Power Manager Telemetry application designed to monitor, analyze, and optimize power consumption across various systems. It collects real-time telemetry data from hardware sensors (CPU, GPU, memory, peripherals), processes it using advanced algorithms and machine learning models to identify patterns, predict usage, and detect inefficiencies. The application provides actionable recommendations to optimize power settings, generates detailed reports and visualizations for insightful analysis, and ensures data security through secure protocols and encryption. Extensive testing scenarios validate its functionality, and the source code repository is accessible for transparency and collaboration, exemplifying a robust implementation of power management standards for enhanced energy efficiency.

**CONTENTS**

1. Introduction ---------------------------------------------------------------------------------------------------
2. Problem Statement ------------------------------------------------------------------------------------------
3. Scope ----------------------------------------------------------------------------------------------------------
4. Infrastructure and Requirements --------------------------------------------------------------------------
5. High-Level Features ----------------------------------------------------------------------------------------
6. Application Workflow --------------------------------------------------------------------------------------
7. High-Level Algorithm --------------------------------------------------------------------------------------
8. Justification for Tools and Technologies Used ----------------------------------------------------------
9. Open Source and System Routines Used -----------------------------------------------------------------
10. Test Plan ------------------------------------------------------------------------------------------------------
11. Learning Outcomes -----------------------------------------------------------------------------------------
12. Result ---------------------------------------------------------------------------------------------------------
13. Conclusion ---------------------------------------------------------------------------------------------------
14. References ---------------------------------------------------------------------------------------------------

**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 and optimizes power consumption. The project requires knowledge of hardware telemetry data collection, power management techniques, and Python system programming.

**Infrastructure Requirements**

**Hardware:**

* Any x86-based Desktop or Server with Windows

**Software:**

* VS Code
* Python
* Docker Desktop

**Requirements:**

* Psutil
* CSV
* JSON
* Threading

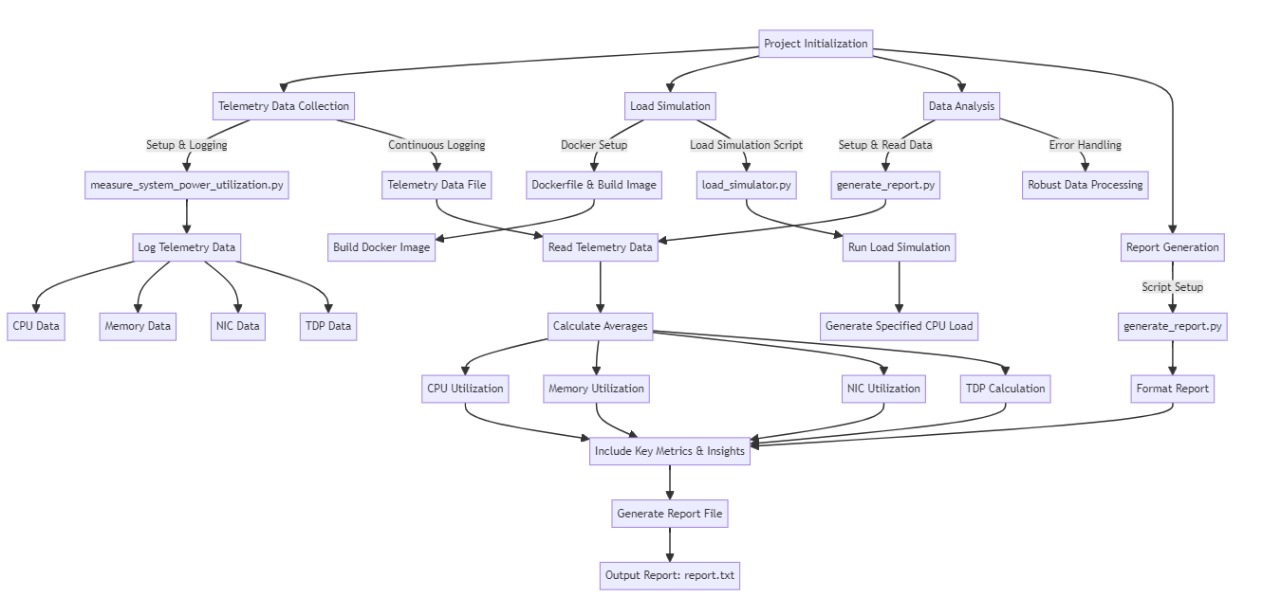
**High-Level Features**

1. Collect real-time telemetry data from hardware sensors (CPU, GPU, memory, peripherals) to monitor power usage.
2. Analyze the collected data using advanced algorithms and machine learning models to identify patterns and inefficiencies.
3. Provide actionable recommendations for optimizing power settings and reducing energy consumption.
4. Generate detailed reports and visualizations for insightful analysis of power usage trends and predictions.

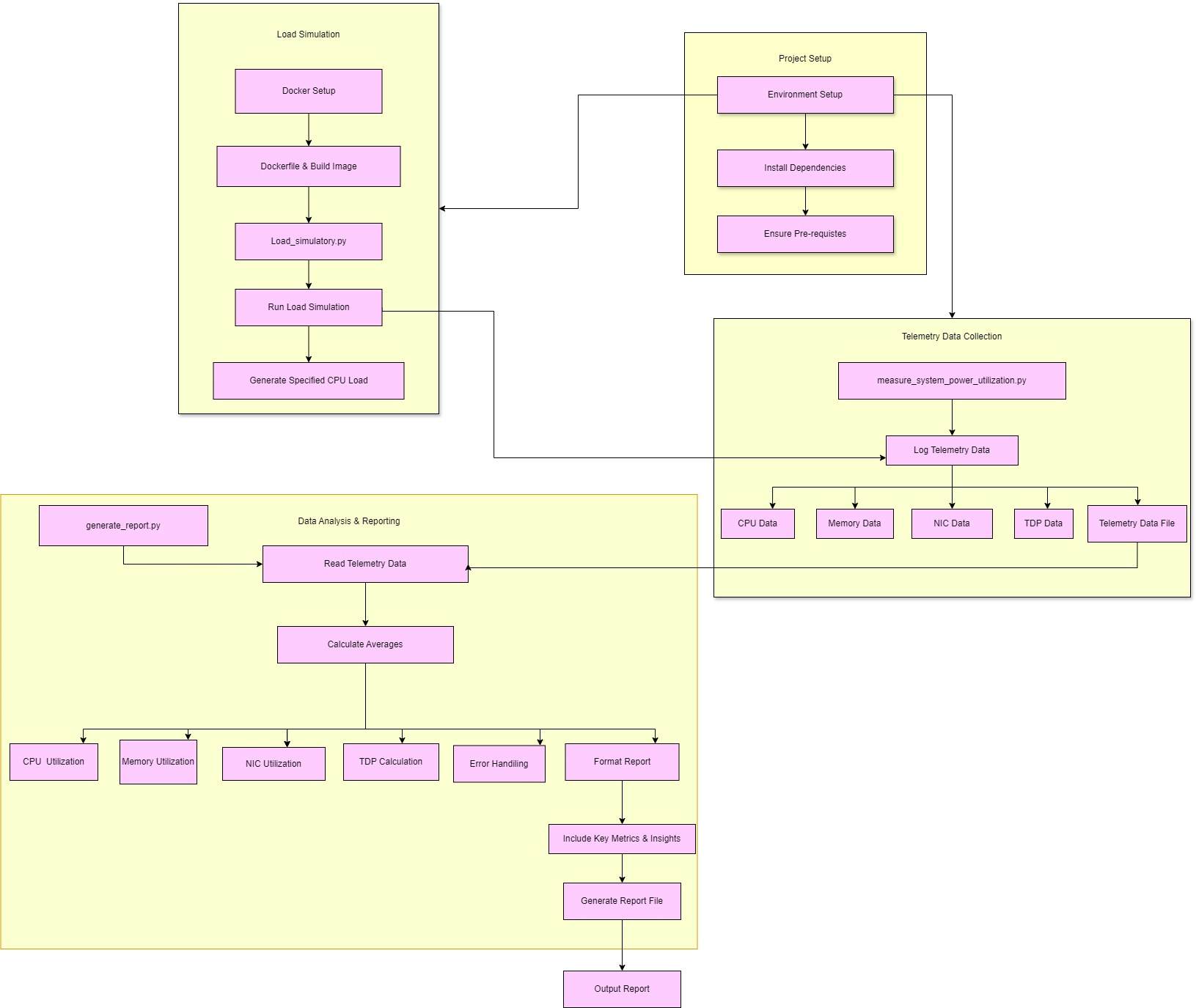
**Application Workflow**

1. **Data Collection:**
   * Continuously collect real-time telemetry data from hardware sensors (CPU, GPU, memory, peripherals).
2. **Data Processing:**
   * Store the collected data in CSV format for further analysis.
   * Use Docker containers to ensure a consistent and isolated environment for processing.
3. **Data Analysis:**
   * Use Python to process and analyze the telemetry data.
   * Apply advanced algorithms and machine learning models to identify patterns, predict usage, and detect inefficiencies.
4. **Optimization Recommendations:**
   * Based on the analysis, provide actionable recommendations for optimizing power settings and reducing energy consumption.
5. **Reporting and Visualization:**
   * Generate detailed reports and visualizations using libraries like matplotlib, pandas, and seaborn.
   * Present insights on power usage trends, predictions, and recommendations for optimization.

**Process Flow**



**Architecture Diagram**



**High-Level Algorithm**

**Data Collection:**

* Continuously gather real-time telemetry data from various hardware sensors (CPU, GPU, memory, peripherals) using Python libraries such as psutil.

**Data Storage:**

* Save the collected telemetry data in CSV format to ensure easy access and analysis.

**Data Analysis:**

* Load the stored telemetry data using pandas.
* Analyze the data to identify power usage patterns and inefficiencies.
* Apply machine learning models using scikit-learn to predict future power consumption and detect anomalies.

**Optimization Recommendations:**

* Based on the analysis, generate actionable recommendations for optimizing power usage, such as adjusting power settings, scheduling tasks, and shutting down idle components.

**Reporting and Visualization:**

* Create detailed reports and visualizations using matplotlib and seaborn to provide insights into power usage trends and predictions.
* Present the recommendations and insights through dashboards and reports for informed decision-making.

**Justification for Tools and Technologies Used**

**Telemetry Data Collection:**

* **Load Simulator:** Utilized to generate simulated telemetry data for testing and analysis. This tool is essential for creating consistent and controlled test scenarios to evaluate the application's performance under different conditions.

**Data Storage and Analysis:**

* **CSV Format:** Selected for data storage due to its simplicity, compatibility, and ease of access for processing and analysis. CSV files are widely used and supported by many data analysis tools.
* **Python:** The primary programming language for developing the application due to its extensive libraries and frameworks for data analysis, visualization, and machine learning.

**Containerization and Environment Management:**

* **Dockerfile:** Used to create a Docker container for the application, ensuring a consistent and isolated environment for running the application. Docker simplifies the deployment process and helps manage dependencies effectively.
* **Docker Desktop:** Provides a user-friendly interface for managing Docker containers on Windows. It ensures a reproducible and consistent environment, making it easier to test and deploy the application across different systems.

**Open Source and System Routines Used**

* **Python:** The primary programming language used for developing the application, leveraging its extensive libraries for data processing, analysis, and machine learning.
* **Dockerfile:** Used to create Docker containers for the application, ensuring a consistent and isolated environment for running, testing, and deploying the application.
* **Load Simulator:** Employed to generate simulated telemetry data, enabling controlled test scenarios and performance evaluation under various conditions.
* **CSV File Operations:** Utilized for handling the input/output operations of telemetry data, ensuring compatibility, simplicity, and ease of access for data analysis.

**Test Plan**

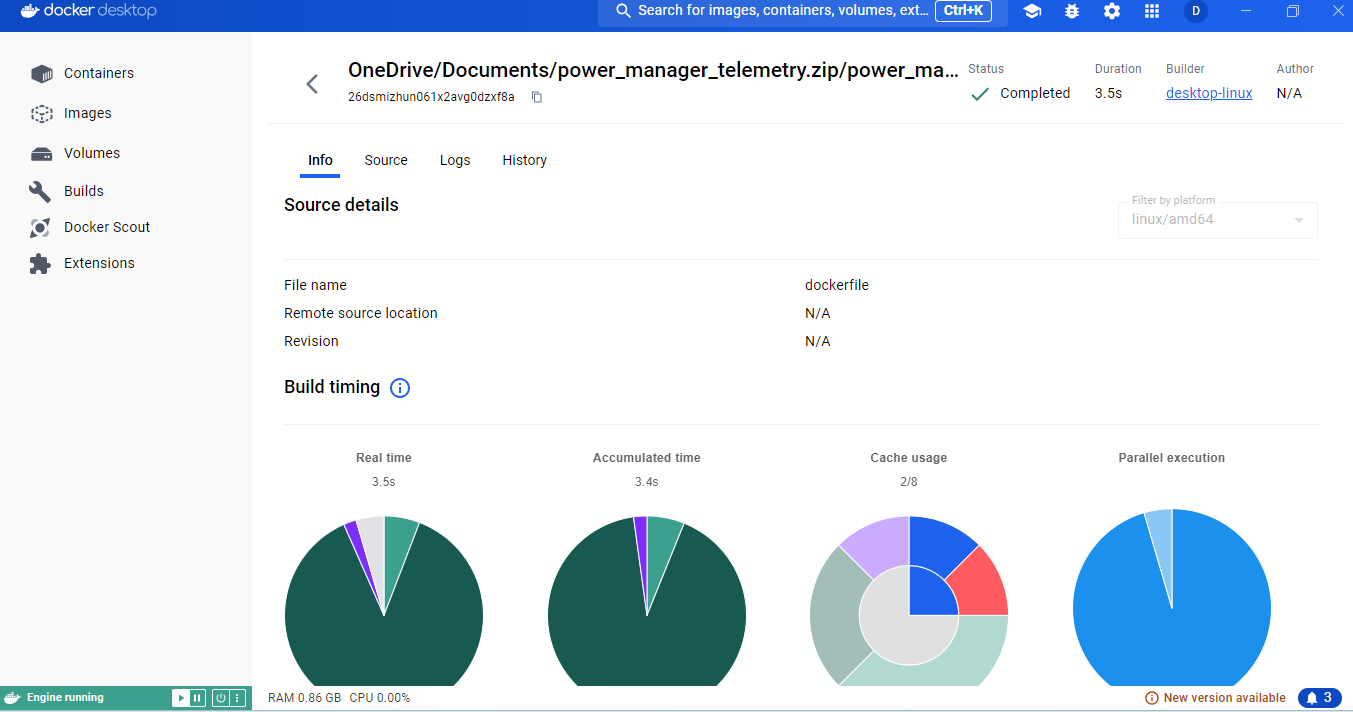
**Test Cases**

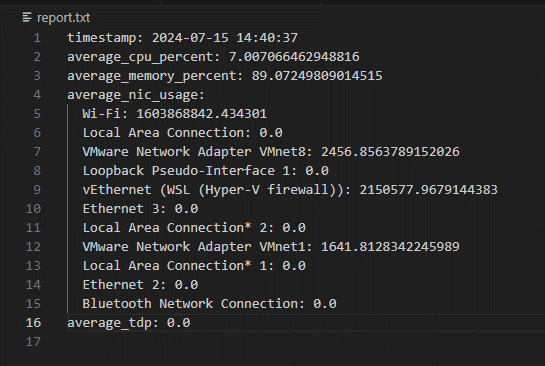
1. **Basic Functionality:**
   * **Collect and Analyze Basic Telemetry Data:**
     + Collect telemetry data from CPU and memory.
     + Verify that the data is accurately collected and stored in CSV format.
   * **Generate Simple Reports:**
     + Create basic visualizations of the collected data.
     + Verify that the reports accurately reflect the telemetry data.
2. **Load Handling:**
   * **Simulate High Load:**
     + Use the load simulator to generate high CPU and memory usage.
     + Verify that the application accurately captures and analyzes the high load data.
   * **Performance Under Load:**
     + Run the application under simulated high load conditions.
     + Verify that the application performs efficiently without significant lag or errors.
3. **Data Analysis and Recommendations:**
   * **Pattern Recognition:**
     + Analyze collected telemetry data to identify usage patterns.
     + Verify that the identified patterns are accurate and relevant.
   * **Optimization Recommendations:**
     + Generate optimization recommendations based on the analysis.
     + Verify that the recommendations are practical and can be implemented..
4. **Security:**
   * **Data Integrity:**
     + Ensure that telemetry data is securely stored and transmitted.
     + Verify that the data remains intact and unaltered during processing.
   * **Access Control:**
     + Implement and test access control measures.
     + Verify that only authorized users can access and modify the telemetry data.

**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. **Data Analysis:** Understand how to analyze telemetry data to find patterns and inefficiencies.
4. **Machine Learning:** Apply machine learning models to predict power usage and detect anomalies.
5. **Docker:** Gain experience with Docker for creating consistent development and deployment environments.
6. **Optimization:** Learn to provide recommendations for optimizing power usage based on data analysis.
7. **Testing:** Use load simulators to test the application's performance under various conditions.
8. **Visualization:** Create reports and visualizations to present power usage insights.
9. **Project Management:** Develop skills in managing and executing a project from start to finish.

**Result**





### Conclusion

This project demonstrates the development of a Power Manager Telemetry application designed to monitor, analyze, and optimize power consumption across various systems. It highlights the importance of collecting real-time telemetry data, using advanced algorithms and machine learning models for analysis, and providing actionable recommendations for power optimization. By ensuring secure data handling and using containerization for consistent environments, this project provides a robust solution for enhancing energy efficiency and system performance.

**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>