Core Requirements

1. GPU Discovery and Inventory:
   - Automatic detection: The tool should be able to automatically discover GPUs in a network or cluster environment.
   - Comprehensive inventory: It should provide detailed information about each GPU, including its model, specifications, and current utilization.
2. Clustering Algorithm:
   - Efficiency: The clustering algorithm should be efficient in grouping GPUs based on factors like performance, compatibility, and workload requirements.
   - Flexibility: It should be customizable to accommodate different clustering strategies and priorities.
3. Tenant Management:
   - Tenant creation: The tool should allow for the creation and management of multiple tenants or users.
   - Resource allocation: It should provide mechanisms for allocating GPU resources to different tenants based on their needs and priorities.
4. Workload Distribution:
   - Intelligent assignment: The tool should be able to intelligently distribute workloads across clusters of GPUs to optimize performance and utilization.
   - Priority management: It should allow for the prioritization of different workloads based on their importance or deadlines.
5. Monitoring and Reporting:
   - Real-time monitoring: The tool should provide real-time monitoring of GPU utilization, workload distribution, and system performance.
   - Detailed reporting: It should generate comprehensive reports on GPU usage, tenant activity, and workload performance.

Additional Requirements

- Security: Implement robust security measures to protect sensitive data and prevent unauthorized access.
- Scalability: Design the tool to handle large-scale GPU clusters and a growing number of tenants.
- Integration: Enable integration with other systems or tools, such as cloud platforms or workload management systems.
- Fault tolerance: Implement mechanisms for detecting and recovering from hardware failures or software errors.

Examples of Use Cases

- Cloud Computing: A cloud service provider can use the tool to efficiently manage and allocate GPU resources to multiple customers.
- Scientific Research: Researchers can use the tool to optimize the execution of computationally intensive simulations and experiments.
- Machine Learning: Machine learning practitioners can use the tool to train and deploy large-scale models on clusters of GPUs.
- Deep Learning: Deep learning researchers can leverage the tool to accelerate the development and training of neural networks.
- Gaming: Gaming platforms can use the tool to manage GPU resources for game servers and ensure optimal performance.

Technical Deliverables

- **Minimum Viable Product : A functional demonstration of the tool's core capabilities, including GPU discovery, clustering, tenant management, and workload distribution.**
- **Source Code: The complete source code for the tool, including any libraries or dependencies.**
- **Documentation: Comprehensive documentation covering the tool's architecture, design decisions, usage instructions, and API reference.**
- **Test Cases: A suite of test cases to verify the tool's functionality, performance, and reliability.**
- **Performance Benchmarks: Results from performance benchmarks to demonstrate the tool's efficiency and scalability.**
- **Security Assessment: A security assessment report outlining any vulnerabilities or risks identified and the measures taken to mitigate them.**
- **User Interface (UI) Mockups: Visual representations of the tool's user interface, demonstrating the intended look and feel.**

Phase	**Activities**
1. Planning and Requirements Gathering	**Define project goals, scope, and requirements. Conduct research and gather user input.**
2. Design and Architecture	**Design the overall architecture, including clustering algorithms, tenant management mechanisms, and user interface.**

3. Development	Implement the core features and optional features as determined by priorities.
4. Testing and Quality Assurance	Develop and execute test cases to ensure the tool's functionality, performance, and reliability.
5. Deployment and Integration	Deploy the tool in a production environment and integrate it with other systems as necessary.
6. Maintenance and Support	Provide ongoing maintenance, support, and updates to the tool.

Project Goals

- Develop a tool that can effectively cluster GPUs and manage their utilization across multiple tenants.
- Ensure the tool is efficient, scalable, secure, and user-friendly.

Project Scope

- Core Features: GPU discovery, clustering, tenant management, workload distribution, monitoring, and reporting.
- Optional Features: Security, scalability, integration, and fault tolerance.