RMM Core Overview

1. Server Component

The server acts as the central command and monitoring hub for all the agents deployed across different systems. Key features for the server include:

- **API Endpoints**: Develop RESTful API endpoints for communication with agents. These endpoints will handle agent registrations, system information and metrics collection, command execution responses, and online/offline status updates.
- **Database**: Implement a database to store information about each agent, including system information, metrics, and activity logs. Consider using a time-series database for metrics for efficient querying over time.
- **Command Dispatch System**: A system to send commands or scripts to agents. This could be based on WebSocket for real-time communication or use a polling mechanism where agents periodically check for commands.
- **Authentication and Security**: Ensure secure communication between agents and the server. Implement authentication for agents and encryption of data in transit and at rest.
- **Dashboard**: Develop a web-based dashboard for real-time monitoring and management. It should display system information, metrics, and online/offline status of agents, and provide an interface to send commands or scripts to agents.

2. Agent Component

Agents are lightweight clients installed on target systems (servers, workstations) that collect data and execute commands from the server. Key functionalities include:

- **System Information Collection**: Implement a method to gather basic system information (e.g., OS type, version, hardware specs) and send it to the server.
- **System Metrics Collection**: Collect system performance metrics (CPU usage, memory usage, disk space, network stats) and report them to the server at regular intervals.
- **Remote Command Execution**: Ability to receive and execute commands or scripts sent from the server and return the output or result back to the server.
- **Online/Offline Status Reporting**: Periodically update the server with the agent's current status. Implement a heartbeat mechanism to detect and report online/offline status.
- **Secure Communication**: Use TLS for encrypted communication with the server. Implement authentication mechanisms to validate the identity of the server and agent.

Additional Basic Functions

- **Logging and Error Reporting**: Implement logging on both the server and agents to record activities, performance metrics, and errors. This can help in debugging and understanding system behavior.
- **Configurable Reporting Intervals**: Allow configuration of how often system metrics are collected and reported to the server.
- **Alerts and Notifications**: Based on the collected metrics and system information, implement a basic alerting mechanism for critical system statuses or metrics thresholds.

Implementation Steps

1. **Define the Data Models**: Start by defining the data models for system information, metrics, and commands.
2. **Setup the Server Framework**: Choose a web framework for Go (e.g., Gin, Echo) to set up the API endpoints and the dashboard.
3. **Implement the Database**: Select and integrate a database that suits your data storage needs (e.g., PostgreSQL for relational data, InfluxDB for time-series data).
4. **Develop the Agent**: Focus on creating a lightweight and secure agent that can perform the required tasks and handle communication effectively.
5. **Security Measures**: Implement authentication, authorization, and encryption mechanisms right from the start to ensure data integrity and confidentiality.
6. **Iterate and Expand**: Start with the basic functionalities and iteratively improve and expand the system based on testing and feedback.