Requirements of the Product

Must Haves:

- The product must have an authorization system that every user has to go through, that gives users access to the product based on their NetID.
- Each user must be able to schedule the usage of the supercomputer with the given faculty information, the number of resources required, the types of resources required and the due date of the request.
- Faculty admins must be able to approve or disapprove the requests that come to their faculty, and if approved, they can choose the date for which it will be processed.
- If a faculty admin approves a request, it must be scheduled before its deadline in the scheduler.
- The requests that reach the last 6 hours before their due date must be assigned to the free resources of the next day (the last day of the request) if there are enough resources left.
- Sysadmins must be able to see the entire schedule of any date, anytime they want.
- The combined resources of the requests scheduled for a certain day should be at most the amount of resources available for that day.
- The schedule for the next day must be finalized the last 5 minutes before the next day.
- The users must be able to have multiple outstanding requests.
- A GPU or memory-intensive request must have at least the same amount of CPU resources requested as well.
- The system must be reached through APIs rather than having a GUI.
- The system must be done using Spring Boot and Gradle.
- The program must be implemented using Java version 11.
- A first fully working version of the system shall be delivered on December 23, 2022.
- The implementation of the application must have 75% of instruction coverage.

Should Haves:

- Faculty admins can approve more requests than what their resources can hold, and if there are enough free resources on the next day, the scheduler should use those free resources for the extra requests.
- Users should be able to add resources as nodes.
- Sysadmins should be able to see the capacity of any of the nodes.
- Users should be able to see the available resources for the next day.
- Users should be able to see the status of requests, so if they are pending (stored, waiting to be accepted or rejected), accepted, rejected or dropped (when a scheduled request is (temporarily) taken out of the schedule because of a decline in available resources).

Could Haves:

- Sysadmins could be able to change schedules based on their desire.
- Sysadmins could be able to allocate resources for each faculty per day.

- Pending requests could be automatically removed from the waiting list when their deadline has passed.

Won't Haves:

- There will not be any GUI implemented.
- There will not be any automatic scheduling algorithm implemented.
- The application will not take care of handling the requests itself.

Color Codes

Functional

Non-functional