# Functional requirements:

## Must have:

- Employees **MUST** be able to submit a request to a faculty which they are assigned to.
- The request **MUST** have a name, description, the amount of resources it wishes to use, and the preferred day to have it run on or before, and the status of the request.
- Employees **MUST** be able to send multiple outstanding requests.
- The faculty account **MUST** review and approve or reject the request.
- The faculty **MUST** have a limited amount of x resources per day.
- Upon approval the faculty **MUST** decide the day the request will be handled on.
- All users of the system **MUST** authenticate themselves to determine who they are and what they can do on the platform
- Users **MUST** have a NetID associated with their account to identify them across the systems of the TU Delft universally
- Users **MUST** have a password that, together with NetID, serves as the credentials for the account
- The NetID **MUST** be a unique string used to identify each user
- Regular users **MUST** only see the available resources for tomorrow

## Should have:

- An employee **SHOULD** be allowed to be assigned to multiple faculties.
- A faculty **SHOULD** be able to decide to release their resources for one or multiple days so others can schedule them in.
- The actual limit of what can be requested for a faculty **SHOULD** be their own daily limit + how many resources are available in the free pool for that day.
- 6 Hours before the day starts, requests **SHOULD** no longer require approval to be run on the supercomputer and requests should automatically be accepted on a first come first serve basis until 0 resources are available.
- 5 minutes before the day starts no requests requesting for that day **SHOULD** be accepted anymore.
- There **SHOULD** be 3 types of resources - CPU, GPU and memory.
- A request for GPU or memory intensive resources **SHOULD** not be allowed to be made without at least an equal number of CPU resources.
- The sysadmins **SHOULD** be able to see the schedules from all days, current capacity of the cluster per node, reserved resources and still available resources for everyday

## Could have:

- A user **COULD** contribute a node (with a name, the url of where to find the node, a token to access the node, and the amount of resources this node will add) of their own to the cluster.
- The users **COULD** take their nodes offline from the cluster. For the nodes it must hold that CPU resources >= GPU/Memory resources. A node shall not be withdrawn the day of the request but should be removed from the cluster starting the next day.
  If the cluster is now too low on compute power, jobs should be dropped and their requesters notified until the required resources can be satisfied.

## Won't have:

- The system **WON'T** have a graphical user interface
- The system **WON'T** be connected to the actual TU Delft authentication system (Single-Sign-On)

## Non-functional requirements:

- The system should be built in such a way that it can be extended with extra functionalities later (modular) and allow for easy integration with other systems (API).
- Individual components of the system need to be scalable so that when the demand for that service increases, the service does not get overloaded (microservices).
- The system should be written in the Java programming language (version 11).
- All interactions with the systems shall be handled through the APIs.
- The system must be built with Spring Boot (Spring framework) and Gradle.
- The password needs to be stored, encrypted, in a safe place