1 Introduction and Use Cases

Your client, HypoSoft, is a midsize software company focused on niche business application development. They develop back-end software for small/medium businesses, such as scheduling for dentists, recordkeeping for drycleaners, and so on. Because of the private nature of their customer data (especially companies in or adjacent to the healthcare market), they host most of their software internally as opposed to using a cloud service. With recent acquisitions, they have multiple datacenters worldwide, each full of racks of servers, networking gear, and more. They currently use a mishmash of spreadsheets to track these systems, but this is cumbersome and error-prone, so they would like a unified system to replace these messy procedures. Further, they would like to gain a better understanding of how their money and rackspace is being used, how to improve physical management of assets, and how to improve efficiency overall. This system will serve the following use cases:

- The system will track data about various models of hardware, as well as instances of those hardware deployed within the datacenter known as assets.
- The system will be able to produce rack elevation diagrams showing deployed equipment in a rack.
- Users will be able to search for and view details of models, assets, and racks.
- Users will be able to bulk-import and bulk-export asset information using a simple text format.
- The system will differentiate multiple datacenter sites.
- The system will map power connections from equipment to Power Distribution Units (PDUs) in racks and allow equipment to be turned on or off by interfacing with the PDUs.
• The system will map physical network connections between equipment and be able to diagram them.

• All users will be able to log in using company single-sign-on. For our purposes, this means support for Duke NetID. Users can be marked as administrators.

• An audit log of all changes and actions undertaken on the system will be maintained.

• To keep access segmented, the system will restrict user actions via a permission system, including per-datacenter permissions.

• Users will be able to print asset barcode labels compliant with a standard label template.

• For real hardware being taken out of service, users will be able to decommission such systems rather than delete them, preserving their state as a historical record.

• Users will be able to make use of a change planner to add, modify, and decommission systems hypothetically before committing the changes; the planner will be able to generate work orders for datacenter staff.

• IT administrators will be able to restore the state of the system from a robust backup system using a clearly documented procedure.

• The system will support modeling blade chassis and blade servers, with some support for power management on blade systems from supported vendors.

• The system will allow individual assets to vary from their published model specifications in order to track on-site hardware upgrades.

• The system will track inventory of non-racked assets held in offline storage sites.

• (OPTIONAL) The system will provide a means for datacenter staff to look up equipment details from a mobile device via asset barcode.

• (OPTIONAL) The team will develop a sales video to potentially market the asset manager software to customers other than HypoSoft.

2 Definitions

1. Model: A kind of IT equipment, defined by fields:
   • Vendor: a short one-line string (e.g. “Dell”), required.
   • Model number: a short one-line string (e.g. “R710”), required.
   • Height: a positive integer number of rack units (U) for the equipment, required, except for blades: blades do not have a height in U, and while blades do come in different sizes, that aspect is not being modeled by this software.
   • Display color: A background color to use when displaying this model in rack elevations. Optional, defaults to a reasonable color. May vary per asset (req 2.2.1.9).
Network ports: a number of ports (usually 1-4 for servers, >24 for networking gear), optional. Blades have no network ports by definition (they connect internally to the chassis).

- Network port names: For each network port, a whitespace-free short string (e.g., “e0”, “e1”, “mgmt”). The names default to simply “1”, “2”, “3”, etc. if omitted.

Power ports: a number of ports (usually 1 or 2), optional. Blades have no power ports by definition (they connect internally to the chassis).

CPU: a short one-line string (e.g. “Intel Xeon E5520 2.2GHz”), optional. May vary per asset (req 2.2.1.9).

Memory: an integer number of GB, optional. May vary per asset (req 2.2.1.9).

Storage: a short one-line string (e.g. “2x500GB SSD RAID1”), optional. May vary per asset (req 2.2.1.9).

Comment: a multi-line string (e.g. “Retired offering, no new purchasing”).

Mount type: Models may be normal rackmount gear, a blade chassis (rackmount and able contain to blades), or a blade (able to be installed within a blade chassis). See definition 15.

2. Asset: A specific piece of owned IT equipment, defined by fields:

- Model: a reference to a model, as described above. Required.
- Hostname: a short string compliant with RFC 1034’s definition of “label” (e.g. “server9”). Optional.
- Datacenter or offline storage site: A reference to a datacenter or offline storage site in which the chosen rack resides. Required.
- Rack Location: The For non-blade assets, this is the rack the equipment is installed in, written as a row letter and rack number, e.g. “B12”. For blade assets, this is the blade chassis asset and slot number into which the blade is installed. Required. Racks in separate datacenters are distinct and unrelated. Not specified for assets stored in offline storage.
- Rack U: An integer indicating the vertical location of the bottom of the equipment (e.g. “5”). Required for non-blade assets. Not specified for assets stored in offline storage.
- Owner: A reference to an existing user on the system who owns this equipment. Optional.
- Comment: a multi-line string (e.g. “Reserved for Palaemon project”). Optional.
- Tags: One or more short plaintext strings associated with this asset. A tag may be shared my many assets (i.e., tags and assets share a many to many relationship). 1
- MAC addresses: 6-byte hexadecimal string per network port, shown canonically in lower case with colon delimiters (e.g., “00:1e:c9:ac:78:aa”), optional.
- Network port connections: For each network port, a reference to another network port on another piece of gear. Optional. Not specified for assets stored in offline storage.

1 Removed 2020-02-10: This requirement has been withdrawn.
• Power connections: For each power port, a choice of PDU in the rack (left or right) and a PDU port number (1..24). Optional. Not specified for assets stored in offline storage.

• Asset number: A six-digit serial number unique associated with an asset; starts at 100000. Generated automatically. Required.

3. **Rack**: A 19” equipment rack deployed in the datacenter. Defined simply by its row letter and rack number, e.g. “B12”. All racks are 42U high. Rack U are numbered 1 through 42, with 1 on the bottom and 42 on top.

![Figure 1: A example rack elevation, truncated.](image)

4. **Rack elevation**: A diagram showing equipment in a rack with accurate vertical positioning and height. Each piece of gear includes vendor, model number, and hostname. See figure 1 for example. For blade chassis, some indication of the blades contained should be shown. At minimum, this could be a count of contained blades, or something more thorough at the developer’s discretion.

5. **Report**: A general term used in database-driven systems; refers to a well-formatted result of a query, usually summarizing a large amount of data at once.

6. **Tag**: A small piece of text that can be attached to assets in the system to help categorize and filter them in different ways.²

7. **Datacenter**: A room or building with rows of racks full of gear (assets). Defined by a short abbreviation (e.g., “RTP1”, max 6 characters) and a longer name (e.g., “Research Triangle Park lab 1”).

8. **PDU**: Power Distribution Unit. The long power strips running down either side of a rack to power the equipment within. Virtually all HypoSoft racks use 24-port PDUs. Some PDUs are controllable over the network, allowing one to turn on and off specific ports remotely. The only model of PDU with remote network control in use by HypoSoft at this time are those managed by a central computer running *PDU Networx 98 Pro* by *Computer Power Corp*. These PDUs were acquired second hand; they were installed in place when HypoSoft acquired their RTP

²Removed 2020-02-10: This requirement has been withdrawn.
The manufacturer, Computer Power Corp, went out of business during the dot-com crash of the early 2000s. As such, no support is available, and no documentation remains of the system.

- The URL for this system is http://hyposoft-mgt.colab.duke.edu:80xx/, where xx is your group number (e.g., group 5 would access port 8005). Do not interact with other groups’ PDU managers! For quick one-off testing, a mock instance is running on port 8000 – this instance can be used by anyone at any time (good for reverse engineering).
- The system does not support authentication; access control is done via software firewall: only certain IP addresses may connect. By default, all Duke IPs in the 152.*.*.* network are allowed; if your system will be connecting from elsewhere, let the instructor know. The mock instance on port 8000 is globally accessible.
- These PDUs are only installed in part of the RTP1 datacenter, hence the naming of the PDUs under management by the system: “hpdu-rtp1-A09L” means “HypoSoft PDU, RTP1 datacenter, rack A9, Left side”.
- Users currently interact with this system manually, and complain of the clumsy interface, especially when power cycling an asset with redundant power (click left PDU, turn off left port, go back, click right PDU, turn off right port, go back, click left PDU again, turn on left port, go back, click right PDU again, turn on right port).
- Meta-note: Computer Power Corp is fictional, but companies using old, unsupported, but functioning gear is very much real. I did my best to give PDU Networx 98 Pro that authentic 90’s web app feel. To aid with debugging, the raw HTTP access log is available via a link at the bottom of the site.\(^3\)

9. **Barcode label**: An adhesive label affixed to assets bearing the text “HypoSoft” and the asset number as both text and a barcode; the barcode format should be Code 128 type C. Labels are 1.75”x0.5” in size and are to be printed on Avery type 5167 label paper. An example mockup of a label is shown below.

![Barcode label](image)

Figure 2: A example barcode label.

10. **Decommission**: To retire an asset from the datacenter while preserving its configuration for future reference. Such assets are removed from the live datacenter and asset listings. This is distinguished from deleting an asset: to delete an asset is to indicate the creation of the asset was in error (i.e., the real equipment never existed); assets that are merely being disposed of for any reason are instead decommissioned. Decommissioned assets do not affect anything else in the system; they effectively do not exist outside of the viewer for such decommissioned assets. As such, they do not affect name collision, deletion warnings, etc.

\(^3\)Side trivia: can you find the hidden easter egg and identify the fictional author of this software?
11. **Change plan**: A named set of hypothetical asset changes; such changes include creating new assets, decommissioning assets, or modifying the hostname, datacenter, rack, rack U, network connections, or power connections of assets. Change plans can also include moves to/from offline storage sites and the handling of blade chassis and blade assets. Change plans cannot include asset deletion, however. Change plans can be created and executed as described in req 10. Change plan names are not unique. Change plans are private to the user who created it.

12. **Executed change plan**: A change plan that has been committed to the actual asset database. For historical reference, these are timestamped, read-only, and are unremovable.

13. **Permission**: An ability on the system that can be conferred upon a user. Users can be given one or more permissions by users that have the administrator permission. Some permissions can be set either globally or on a per-datacenter basis. The ability of logged-in users to view (but not modify) inventory information requires no explicit permission. The grantable permissions are:

   - **Model management permission**: Allows creation, modification, and deletion of models.
   - **Asset management permission**: Allows creation, modification, decommissioning, and deletion of assets. May be conferred globally or per-datacenter per-site (where site is either a datacenter or offline storage site). Naturally, if a user has a per-datacenter per-site asset management permission, then they should only be able to affect assets in that datacenter site. When moving assets between datacenters sites, such users should only be able to move assets among datacenters sites they have permission on.
   - **Power permission**: Allows power control of assets for users that are not the explicit owners of the asset in question. Naturally, this only affects assets connected to network-enabled PDUs or blade assets in blade chassis from vendor “BMI”.
   - **Audit permission**: Allows reading of the audit log.
   - **Administrator permission**: Inherits all of the abilities described above. Can also confer or revoke permissions onto users (per req 1.10).

14. **Administrator**: A user with administrator permission.

15. **Blade chassis and blades**: Some servers and network gear come in a modular form factor called blades. In this design, a rackmount blade chassis is installed; it will have a number of slots to accommodate easily installable gear called blades inside of it. For servers, these are commonly called blade servers. The blade chassis will handle management and networking for a number of servers, allowing a much higher density of compute resources in a given amount of rack space, reducing the labor for installation, and reducing the number of discrete management portals an administrator has to worry about. Take note:

   - Individual blades will be modeled as having no network or power ports, as these connections are made internally within the chassis. For network neighborhood graph purposes, blades are implicitly connected to their housing blade chassis. The chassis itself will have the usual network and power connections.
The blade approach is also common in networking gear, but with different nomenclature. Chassis are referred to as *director-class switches* with the units that go inside being referred to as *line cards*. For our purposes, we'll just use the “blade chassis”/“blade” terminology for all such gear.

Blade chassis have a number of slots. For our purposes, we will not model the dimensions and restrictions of what blade can go into what chassis and in what quantity; the software will assume that any chassis can hold up to 14 blades of any type in slots numbered 1-14. A chassis slot can hold exactly one blade, and could be empty.

Examples of blade chassis designs for servers include the IBM BladeCenter (shown below), Lenovo Flex System, Dell MX series, Cisco UCS, and for our purposes, the fictional *BMI BladeChassis*. Examples of blade chassis designs for networking include the Cisco 6509, the Juniper MX series, and the HPE 10500 series. An IBM BladeCenter H is shown below as an example.

![Figure 3: An IBM BladeCenter H housing 14 blades in 9U of rack height.](image)

16. **BMI BladeChassis**: A fictional blade server system developed by “BMI Corporation”. Such chassis hold up to 14 blades. Power for such blades can be managed via SSH command line via a system known as *BCMAN*. Similar to *PDU Networx 98 Pro*, there is no formal documentation available, but *BCMAN* does have built-in help. Note the following:

- *BCMAN* should be applicable to all blades installed in a blade chassis made by vendor “BMI”.
- *BCMAN* is available at hyposoft-mgt.colab.duke.edu port 2222. The username is admin#, where # is your group number. Passwords will be distributed separately.
- In real life, such a system would need to connect to the actual chassis by hostname or IP address. For our purposes, it will accept any legal hostname provided and simulate there being 14 valid blades installed, all powered on by default.
- Each user account manages a separate universe of chassis and blades (e.g., the actions of admin3 will not effect the view shown to admin5).
- Please report any issues found with the system to the instructor.
- Meta-note: Like *PDU Networx 98 Pro*, *BCMAN* is fictional, but it is very similar to the command-line interface of the IBM BladeCenter management system.
The following command-line session shows the user turning off blade 4 in chassis “mychassis1”:

![Command-line session]

Figure 4: A example *BCMAN* session.

17. **Offline storage site**: A location where assets can be staged outside of racks. These are similar to datacenters in several ways:

- They are defined by a short abbreviation (e.g., “RTPB1S”, max 6 characters) and a longer name (e.g., “Research Triangle Park building 1 storage”).
- For organization purposes, offline storage sites will function like datacenters, except assets in offline storage have no rack location or connections of any kind.
- Offline storage sites are governed by permissions in a manner similar to datacenters (i.e., one can have asset management permission globally or for a single offline storage site).
In practice, such sites are usually the staging ground for newly arrived equipment as it is inventoried and given asset numbers prior to installation into a proper datacenter.

3 Requirements

A note on requirements: No set of requirements is perfect, and that is certainly true here. I’m sure that contradictions, under-specified behavior, and unintended consequences will be revealed. Your overriding goal should be to produce a quality system; if you believe that goal would be better served if a requirement were altered or interpreted a certain way, ask about it, and get the conclusion in writing. The result may be a variance in a requirement for a specific team, or even modification of this requirements document for all teams. In short, if unsure, ask.

Some requirements have attached an informal tip, clarification, or example – these do not alter the requirements themselves, but are meant to answer likely questions about a requirement.

1. Server

1.1. Your software must have a server that supports an arbitrary number of users.

1.2. During the install/setup process, a special user named “admin” is configured.

1.3. Users are defined by username (e.g. “tbletsch”), display name (e.g. “Tyler Bletsch”), and email address (e.g. “Tyler.Bletsch@duke.edu”), and password. The system shall allow the use of the Duke NetID system to allow all users to login using their Duke credentials in addition to supporting locally created users. The special local “admin” account remains, and has administrator permission. For NetID users, the NetID is their effective username on the system. NetID users new to the system have no special permissions by default and are therefore “normal users”.

1.4. A user accessing the system prior to logging in should be able to access nothing but a login prompt. Login is via username and password.

1.5. Any stored passwords must be kept in a secure manner (i.e., salted + hashed at minimum)

1.6. All communication between the clients and server must be encrypted.

   Tip: For web-based solutions, this means using HTTPS.

1.7. The server must maintain state in a persistent fashion.

   Tip: For web-based solutions, this just means using a database or similar.

1.8. For all views which show a potentially unbounded number of records, the response time of the interface shall not depend on the quantity of records unless a full listing is explicitly requested by the user.

   Tip: This implies some form of pagination so that only a finite number of records are retrieved at a time. Pagination can be explicit (page 1 of N) or implicit (infinite scrolling with auto-loading). The latter part of the requirement (“unless a full listing is requested”) implies a “show all” button or similar. Other UI solutions are likely also possible.

1.9. A user input is said to be assisted if it is a user-selected reference to an existing record (model, asset, etc.) where the UI provides a listing, inline search, autocomplete, and/or other means to allow easy and efficient selection. Unless otherwise specified in this document, all selections of an existing record should be assisted.
1.10. The system shall track permission level for each user as described in definition 13. The special local “admin” user has permanent implicit administrator permission. A user that logs in by NetID for the first time or is created by an administrator has no explicit permissions; this is a “normal user”. A user may not do anything that exceeds their permissions as described in definition 13.

Note: Every effort has been made so that these requirements reflect the permissions described. Please notify the instructor of any discrepancies found. The administrator permission subsumes the others, so if another requirement specifies a certain permission is needed, an administrator can implicitly also do that task. Also, if a requirement says a certain permission can do something and a sub-requirement simply says that the “user” will do part of that thing, then the word “user” in that context is simply a pronoun referring to the initiating user who has the appropriate permission, not just any normal user.

1.11. Administrators can grant or revoke permissions to any existing user (either NetID-based users or local users).

1.12. Administrators can create and delete “local” (non-NetID) user accounts.

2. Asset data management

2.1. **Model management**: Users with model management permission shall be able to add, modify, and remove models within the system; users shall be able to review them.

2.1.1. Models are to be created and modified with the fields described in definition 1, subject to these additional constraints:

2.1.1.1. Vendor input should be assisted based on past-seen vendors, such as with an auto-complete system.

2.1.1.2. The combination of vendor and model number must be unique.

2.1.1.3. It should be possible to specify network ports either as a simple count (e.g. 3 ports implicitly named “1”, “2”, and “3”) or explicitly with names (e.g. “e0”, “e1”, and “mgmt”).

2.1.2. Users with model management permission seeking to remove a model shall only be able to do so if no assets of the model exist. Even in this case, a confirmation dialog shall be shown.

2.1.3. Users may view a table of stored models showing all the short-form fields.

2.1.3.1. The view should be sortable by all shown fields.

2.1.3.2. It should be possible to filter this view by keyword on all text fields and by range selection of numeric fields.

Tip: Think the filter options on a web store such as Amazon.

2.1.3.3. Users should be able to navigate from this to a detail view (see req 2.1.4).

2.1.4. Users may view a detail view of a model showing all fields. This view should also list the assets that exist of this model and allow navigation to an asset’s detail view (req 2.2.4).

2.2. **Asset management**: Users with asset management permission shall be able to add, modify, and remove assets within the system; users shall be able to review them. For users whose asset management permission is tied to one or more datacenters, modifications are restricted to those datacenters.
2.2.1. Assets are to be created and modified with the fields described in definition 2, subject to these additional constraints:

2.2.1.1. Model input should be very assisted, with both autocomplete and picklist.

2.2.1.2. Rack number must be among those created in the system for the chosen data-center (see req 2.3). Alternately, an offline storage site could be chosen instead (req 12). Blades do not have a rack, and instead reside within a blade chassis in a particular slot (see req 2.2.1.10).

2.2.1.3. The rack U must fit logically in the rack: it may not conflict with other gear and it must fit within the confines of the rack. In the event of a conflict, the exact reason must be shown, including the specific conflicting asset(s).

Example: If a 4U server is installed at U 5, then it occupies U \{5,6,7,8\}, so it would conflict with anything racked in those positions, as well as a 3U piece of gear racked at U 3 (as that occupies U \{3,4,5\}). A 4U server racked at U 40 would also fail, as the rack height is only 42.

2.2.1.4. For tags, the user should be able to easily assign existing tags as well as create new tags on the fly.\(^4\)

2.2.1.5. For MAC addresses, the system should accept a six-byte hexadecimal value with any byte separator punctuation (including colon, dash, underscore, and no separator at all). Upon accepting the value, it should be formatted into a lower-case colon-delimited canonical form.

2.2.1.6. For network ports, a specific network port on another piece of existing hardware is to be selected. Steps should be taken to make this as efficient as possible. Note that connections are symmetric (if A connects to B, then B connects to A), and a port can go at most one place (so A cannot be connected to both B and C). If an attempted connection is invalid because the target port is already connected to something else, a clear message indicating this should be shown, including what the conflicting device is (e.g., “can’t connect host1 port e1 to switch1 port 22; that port is already connected to host5 port e1”). Assets in offline storage have no network connections. Blade network connections are not modeled in this software except for network neighborhood graph purposes, wherein blades are implicitly connected to their housing blade chassis. Blade chassis network connections are modeled normally.

2.2.1.7. For power ports, the user should be able to quickly pick “left” or “right” PDU as well as a PDU port number. As with network ports, power ports cannot be doubly connected, and appropriate error messaging should be displayed. The system should employ sensible defaults during the input process, if the user elects to set up power connections\(^5\) : for systems with 2 power ports, port 1 should default to the left PDU and its first free port; port 2 should default to the right PDU and the same PDU port number as was chosen for port 1 (as power connections are usually made symmetrically). Assets in offline storage

\(^4\)Removed 2020-02-10: This requirement has been withdrawn.

\(^5\)Clarified 2020-02-02. Defaults here refer to the input fields the user manipulates within the interactive UI, not fields set persistently automatically at asset creation. Should the user not elect to set up power connections, the default should be “no connection”.

Updated March 31, 2020
have no power connections. Blade power connections are not modeled in this software (but blade chassis power connections are).

2.2.1.8. For asset number, a six-digit number greater than 100000 should be automatically generated by default, but it should be possible for the user to override this, provided they choose an unused (i.e., unique) value that is also compliant.

2.2.1.9. Certain properties derived from an asset’s model may be customized per-asset (display color, CPU, memory, and storage). The interface must make it clear when such a field has been modified from the model-level default. During editing, it must be possible to revert such fields to their model-level defaults.

2.2.1.10. Blade assets should have their location specified as containing blade chassis and slot number. In editing a blade’s location, the interface should allow selection of an existing chassis, then show the slots taken and free in the chosen chassis to allow the blade to be installed in a specified free location.

2.2.2. Users with asset management permission seeking to remove an asset will be prompted with a confirmation dialog first. Users should not be able to remove a blade chassis that contains blades; an appropriate error should be shown in this case.

2.2.3. Users may view a table of stored assets showing all the short-form fields excluding assets in offline storage; those are viewed elsewhere (req 12.2).

2.2.3.1. The view should be sortable by all shown fields. There should be a combined sort for rack number and rack U (e.g. “A1 U1, A1 U2, ..., A2 U1, A2 U2, ...”)

2.2.3.2. It should be possible to filter this view by keyword on model and/or hostname as well as by a range of racks in a given datacenter.

2.2.3.3. Users should be able to navigate from this to a detail view of an asset (see req 2.2.4).

2.2.3.4. The view should be filterable by an easily selected set of tags; items shown must have all selected tags.6

2.2.3.5. Blade chassis and blades should be navigable from this interface, but given the unique parent-child relationship of these assets, developers have discretion in how best to represent this.

2.2.4. Users may view a detail view of an asset showing all fields. This view should allow navigation to its model’s detail view (req 2.1.4).

2.2.4.1. From the detail view, users should be able to view network connected devices. This listing should allow direct navigation to the detail view for such connected devices, e.g. via hyperlink.

2.2.4.2. On the detail view, an option should be available to show a “network neighborhood”: a visual graph of devices and their connections for the asset, all assets connected directly, and all assets connected to those (i.e. assets at graph distance 0..2). See figure 6 (page 21) for an example of one possible implementation of such a feature. It is possible to combine this with req 2.2.4.1. For graph purposes, blade chassis are implicitly connected to their housed blades.

2.2.4.3. In concert with the power management features described in req 6, it should be possible for the user to view the on/off state of the asset’s power ports, provided

6Removed 2020-02-10: This requirement has been withdrawn.
the asset is connected to managed PDUs or is a blade in a blade chassis from vendor “BMI”. Given that this has to query PDU Networx 98 Pro or BCMAN, this should only be done in the detail view rather than en masse, and it is acceptable if the user has to interact with the UI to issue this query.

2.2.4.4. When viewing blade asset, a view of the its chassis’s installed blades should be included, with the viewed blade highlighted. It should be possible to navigate to the chassis’s detail view from here.

2.2.4.5. When viewing blade chassis asset, a view of the its installed blades should be included. It should be possible to navigate to any individual blade’s detail view from here.

2.2.4.6. The view should show fields derived from an asset’s model (e.g., height, CPU, etc.). Such fields that have been edited specifically within this asset (per req 2.2.1.9) should be visually distinct to indicate that this asset differs from its base model.

2.2.5. Users with asset management permission shall be able to decommission an asset. This action removes it from the live database, but preserves it in a read-only state for historical reference. See req 11. Decommissioning an asset requires asset management privilege, and the user must confirm the action. This can be done on assets in a datacenter or in offline storage. Decommissioning a blade chassis implicitly decommissions all of its blades as well; the confirmation warning should indicate this if applicable.

2.2.6. Users with asset management permission shall be able to move an asset into an offline storage site (provided their asset management permission extends to both the current datacenter and the destination offline storage site). This action discards certain fields such as network and power connections, and therefore a confirmation warning should be shown first. Moving a blade chassis to offsite storage implicitly moves all of its blades as well; the confirmation warning should indicate this if applicable. Developers have their choice of how to handle blades in offline storage (either making them live in a blade chassis as normal, making them live outside a blade chassis, or allowing either behavior). See req 12 for details on offline storage.

2.3. Datacenter management: Users with asset management permission shall be able to add, modify, and remove datacenters and racks within the system; users shall be able to review them. Adding, modifying, and removing datacenters requires global asset management permission. Adding, modifying, and removing racks in a datacenter requires asset management permission for that datacenter (or global asset management permission).

2.3.1. In a given datacenter, racks are to be created en masse using a range system: by specifying the row letter span and rack number span to create at once. For example, it should be possible to create rows A-E each with racks 1-20 (i.e. 100 racks) in a single step. Removing racks should function similarly.

2.3.2. Administrators seeking to remove a rack shall only be able to do so if no assets are installed in it. Even in this case, a confirmation dialog shall be shown.

2.3.3. Users should be able to select a range of rows and rack numbers to view; the racks as should be presented as compact rack elevation diagrams (per definition 4). It should
be possible to navigate to a particular asset’s detail view from here (req 2.2.4). For blade chassis, some indication of the blades contained should be shown. At minimum, this could be a count of contained blades, or something more thorough at the developer’s discretion.

2.3.3.1. The rack elevation view should be printable, such that multiple racks can be shown in landscape in their entirety (see figure 5, page 20).

2.3.4. Administrators should be able to view all datacenters, rename their abbreviations and display names, create new datacenters. Administrators should be able to remove datacenters that have no racks.

2.4. **Tag management**: In addition to the in-line tag management described, administrators should be able to view all tags, rename them, create new tags, and remove them at will. Tags that are attached to one or more assets should present a warning before removal. 7

3. Bulk import/export facility

3.1. Users with asset management appropriate permission shall be able to import new models and assets into the system by means of a format compatible with modern spreadsheet software (CSV, XLSX, or similar). The customer is accepting proposals on the format. Importing models requires model management permission. Importing assets requires asset management permission, and for users whose asset management permission is tied to one or more datacenters, modifications are restricted to those datacenters.

3.2. The import interface shall include documentation as to the import format.

3.3. The import action shall only occur if the entire input is free of name conflicts or otherwise problematic issues; if such issues arise, the precise nature of the error should be presented to the user in enough detail that it can be corrected.

3.4. If an import contains identical record(s) to those already in the system, such records should be ignored.

3.5. If an import contains record(s) that match on vendor and model number (for models) or hostname (for assets), the user should be warned about all such records in detail, and if the user approves, the records should be modified to match the imported data.

3.6. After a successful import, a count and list of records that were added, updated, and ignored should be provided.

3.7. The system shall be able to export any of the above data in a format compatible with import. The specific records exported should be filterable by keyword on model and/or hostname as well as by a range of racks. Exporting does not require asset management permission.

    *Note: This allows for an export/modify/import workflow when large-scale changes are needed.*

3.8. It should be possible to import assets directly into an offline storage site and to export assets currently held in offline storage.

---

7 Removed 2020-02-10: This requirement has been withdrawn.
8 Corrected 2020-03-02
9 Added 2020-03-02
Suggestion: One way to represent this is by analogy: datacenter → offline storage site. Also, note that as a natural consequence of this requirement, it should be possible to move assets to/from offline storage using the export/modify/import workflow.

3.9. Blades and blade chassis should be supported by the import facility.

Suggestion: Similar to above, one might represent blade location by analogy: rack → chassis hostname, rack location → slot.

4. Reporting

4.1. **Rack usage report**: Users shall be able to create a tabular report the percentage of rackspace free versus used, allocated per vendor, allocated per model, and allocated per owner. It should be possible to produce this report globally as well as per datacenter.

5. Documentation

5.1. **Developer guide**: A document shall be provided which orients a new developer to how your system is constructed at a high level, what technologies are in use, how to configure a development/build environment, and how the database schema (or equivalent) is laid out.

5.2. **Deployment guide**: A document shall be provided which describes how to install your software entirely from scratch. It should start by describing the platform prerequisites (e.g., Linux distro, required packages, etc.), then mechanically describe every step to deploying your system to production readiness.

5.2.1. In addition to covering how to install the system with “stock” default data, the procedure to install the system from scratch using backed up data should also be included (i.e., disaster recovery).

5.3. **Backup admin guide**: A document shall be provided which explains the backup solution so that a system administrator unfamiliar with your software could configure it from scratch, restore the database to any given backup, and test a backup for validity. See req 9.

6. **Power management**: An asset’s owner or a user with power permission may power a piece of equipment on and off using network managed PDUs or blade chassis, provided the asset either has power connections noted in the system and that these power connections go to PDUs that are network-manageable (i.e., managed by *PDU Networx 98 Pro* or is a blade housed in a blade chassis manufactured by vendor “BMI”; see definition 8 and 16).

6.1. If an asset qualifies as power manageable by the system and the user is authorized, controls should be visible on the asset list and detail view to power on, power off, and power cycle it.

6.2. Upon choosing a power option and approving a confirmation dialog, the system should issue the appropriate POST request to *PDU Networx 98 Pro* for all power port(s) of the asset or the appropriate SSH command sequence for *BCMAN*. For power cycling, the system should power the port(s) off, wait 2 seconds, then power the port(s) on.

6.3. Power state for an asset should be visible as described in req 2.2.4.3.
6.4. The system must be resilient in the face of connectivity issues to *PDU Networx 98 Pro* or *BCMAN*, i.e. showing a well formatted error message when a power-related request cannot be completed.

7. **Global system logging**

7.1. The system shall record a log of all actions undertaken in the system (i.e., any action that alters the system state, including creation/modification/deletion of user, model, asset, and related data, as well as power operations. Log entries shall include the initiating user, the entities involved, the nature of the event, and the time and date.

7.2. Users with audit permission shall be able to view this log.

7.3. The log view should allow searching by user or asset.

7.4. Users consulting the log shall be able to navigate directly from a reference to a model or asset to the relevant detailed view (see reqs 2.1.4 and 2.2.4).

7.5. Users should not be able to tamper with the log in any way, regardless of permission.

7.6. Execution of change plans should be explicitly noted in the log; merely noting the effects of the change plan is not sufficient. Creation and management of change plans need not be logged, just their execution.

8. **Asset label generation**

8.1. Users will be able to apply filters as in req 2.2.3.2 to identify a set of assets, then be able to select a user-specified subset of those assets. This latter selection process will include “select all” and “select none” options to expediate selection. It should be possible to build up a set of selections over multiple filters and selection steps (e.g., filtering to RTP1 rack A1 and choosing all, then filtering to RTP1 rack A2 and adding a specific handful of additional assets).

8.2. For the selected assets, the system must be able to print a set of barcode asset labels as described in definition 9 for use with Avery type 5167 label paper. Labels should be laid out efficiently using the minimum number of pages possible.

9. **Backups**: You must deploy a backup solution for your system’s database.

9.1. Backups shall be automatic and taken daily.

9.2. Backups shall be kept with a staggered retention (7 daily backups, 4 weekly backups, 12 monthly backups).

9.3. Backups must be stored on a separate system.

9.4. The backup system must require separate credentials to access.

9.5. The backup system should report on progress and alert on failure; this could be via email or another directed communication mechanism.

9.6. The backup system may be built either out-of-band from the main software (e.g. a background database dump restored manually by a sysadmin) or in-band (e.g. the software itself exporting its database using internal automation). If it is in-band, all backup operations should be restricted to administrators.
10. **Change planner system**

10.1. Users with asset management permission will be able to access a change planner system. For users whose asset management permission is constrained to one or more datacenters, only changes affecting such datacenters can be input into a change plan.

10.2. In this system, users will be able to create change plans (per definition 11). The user will provide a name for the plan, and then be able to make proposed asset changes. The interface to do so will be similar to the usual interface for directly creating, modifying, and **decommissioning** assets, but will make it visually clear to the user that such changes are just being made within the hypothetical change plan.

10.3. Users will be able to browse a list of their own change plans, but not those of others.

10.4. Users will be able to enter a detail view for a change plan that succinctly and clearly summarizes all effects it will have on assets (e.g., “decommission host ‘alpha1’, remove cable on ‘alpha2’ port ‘e1’, replace connection on ‘alpha2’ port ‘e2’ to ‘dc-core-sw1’ port ‘22’, add new asset 100404 as host ‘alpha3’ with following configuration ...”).

10.5. As the actual asset data evolves, it is possible for a change plan to become **conflicted** (e.g., if a destination rack location becomes populated or if an asset slated to be edited in a change plan is decommissioned in the real environment). Such conflicts shall be presented in a visually distinct way with enough supplemental detail to understand them in the detail view and editing interface. Note: a change plan can only be in conflict with the live asset data; it is not necessary to track conflicts between two different non-executed change plans.

10.6. Users shall be able to edit a change plan in a similar manner to its original creation. This editing mode should make it easy to identify and resolve conflicts (e.g., noting that a destination rack location is populated and aiding the user in selecting a free one).

10.7. For a given change plan that is without conflict, users shall be able to generate a printable work order that clearly and succinctly explains to datacenter staff the installations, deinstallations, and cabling steps needed to affect the changes specified in the plan. This plan should contain all information needed for a worker with basic IT datacenter knowledge to perform the physical steps needed.

10.8. Users shall be able to request that a given change plan be **executed**, i.e. that those changes are being made in real life. In this case, the user should be presented with the list of effects it will have on assets (as in req 10.4) and asked to confirm. Upon confirmation, the changes listed shall be made to the real asset database.

10.9. Executed change plans become marked as such and annotated with the time and date of execution. These can no longer be edited.

10.10. Users may delete or rename non-executed change plans.

10.11. With regard to asset numbers, autogeneration of asset numbers should be deferred until execution of a change plan, but manually entered asset numbers should still be accepted during creation/editing of the change plan.

10.12. **Change plans should be able to incorporate per-asset customized fields (req 2.2.1.9).**
10.13. Change plans should be able to move assets to/from offline storage (req 12).

11. **Decommissioning assets**

11.1. As specified in req 2.2.5, it is possible for a user to decommission an asset. As such, users should be able to review a listing of decommissioned assets with an interface similar to the live asset view described in req 2.2.3.

11.2. In addition to the usual fields, the timestamp and user who performed the decommissioning should be shown.

11.3. The view should allow for filtering by decommissioning user and bounds of time of the event (e.g. “before 6/1/2019”, “between 1/1/2020 and 1/31/2020”, etc.).

11.4. Users should be able to navigate to a detail view for a decommissioned asset, which should function similar to that described in req 2.2.4, but with addition of the decommissioning info described above.

11.5. The information shown should permanently reflect the state of the asset at the moment it was decommissioned. For example, the network neighborhood view should show neighboring assets from the moment of decommissioning regardless of what happens in the live database after the fact.

12. **Offline storage**

12.1. Users with global asset management permission shall be able to add, modify, and remove offline storage sites within the system; users shall be able to review them. It should not be possible to remove offline storage sites that contain assets.

12.2. Users should be able to review a listing of assets stored in an offline storage site with an interface similar to the live asset view described in req 2.2.3.

12.3. Users should be able to navigate to a detail view for a stored asset, which should function similar to that described in req 2.2.4.

12.4. As specified in req 2.2.6 and req 10.13, it is possible for a user to move assets from a datacenter to offline storage. Further, users with either global asset management permission or asset management permission on an offline storage site and another site (either offline or datacenter) should be able to move assets between the two. If moving rackmount gear from offline storage to a datacenter, the rack location should be specified as part of the move. If moving a blade, the blade chassis and slot into which it will be installed should be specified.

Note: In other words, any asset should be able to move to any site, provided the user has asset management permission on source and destination sites; assets heading to datacenters need all appropriate location info specified.

13. **Mobile barcode lookup** (OPTIONAL: This requirement may be pursued for extra credit equal to approximately +15% of the test plan score.)

13.1. The system will furnish some means for a mobile device (Android phone/tablet, iPhone, or iPad) to scan an asset barcode label and display all pertinent information about the asset. This view is simply read-only.
13.2. For grading purposes, a method of testing that does not require any specified model of
hardware is needed. This can be achieved by either (a) implementing the feature as a
mobile web interface or (b) implementing a native app and providing a cross-platform
simulator (supporting Windows and Mac) and documentation for same. For mobile
web solutions, you can assume a reasonably modern mobile browser (e.g., Chrome for
Android v80).

13.3. Partial credit will be provided to solutions that function without authentication; full
credit for solutions that have the same authentication methods as the primary applica-
tion.

14. **Marketing video** (OPTIONAL: This requirement may be pursued for extra credit equal to
approximately +15% of the test plan score.)

14.1. Based on a recent meeting with a NetApp representative, your company leadership has
decided that it may be feasible to market this software to customers other than HypoSoft.
As your company is a small start-up, you do not have a formal marketing team, so your
group has been asked to develop a 4-6 minute sales video to kickstart your effort.

14.2. Points will be awarded for professionalism, succintly capturing your value proposition,
clearly differentiating from competitors, and overall attractiveness of visual aesthetic. A
minority of points will be awarded competitively.

14.3. Submission of this component will be via YouTube link (either public or unlisted) sub-
mitted alongside your documentation materials.
Figure 5: A example rack elevation made compactly printable.
Figure 6: A example network neighborhood graph. In this case, the asset “rtp-dfm-sv1” was selected. Note: many visual implementations are possible; this is just one idea.