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. This means they have a small datacenter full of racks of servers, networking gear, and more. 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 and 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.

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.
   • Display color: A background color to use when displaying this model in rack elevations. Optional, defaults to a reasonable color.
   • Ethernet Network1 ports: a number of ports (usually 1-4 for servers, >24 for networking gear), optional.
     - 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.
   • CPU: a short one-line string (e.g. “Intel Xeon E5520 2.2GHz”), optional.
   • Memory: an integer number of GB, optional.
   • Storage: a short one-line string (e.g. “2x500GB SSD RAID1”), optional.
   • Comment: a multi-line string (e.g. “Retired offering, no new purchasing”).

2. Instance Asset2: 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”). Required. Optional.
   • Datacenter: A reference to a datacenter in which the chosen rack resides. Required.
   • Rack: The rack the equipment is installed in, written as a row letter and rack number, e.g. “B12”. Required. Racks in separate datacenters are distinct and unrelated.
   • Rack U: An integer indicating the vertical location of the bottom of the equipment (e.g. “5”). Required.

---

1This more generic name allows us to represent alternative kinds of networking.
2This term was previously labeled an “instance”, but the customer is moving away from this word choice; the term “asset” should be used instead. This change is reflected in the document, and for readability, the renaming is not marked by the usual cross out and blue text.
• 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 with many assets (i.e., tags and assets share a many-to-many relationship).\(^3\)
• MAC addresses: 6-byte hexadecimal string per Ethernet network port, shown canonically in lower case with colon delimiters (e.g., “00:1e:ac:78:aa”), optional.
• Network port connections: For each network port, a reference to another network port on another piece of gear. Optional.
• Power connections: For each power port, a choice of PDU in the rack (left or right) and a PDU port number (1..24). Optional.
• 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.

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.\(^4\)

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”).

---

\(^3\)Removed 2020-02-10: This requirement has been withdrawn.
\(^4\)Removed 2020-02-10: This requirement has been withdrawn.
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 datacenter. 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.\(^5\)

### 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**

---

\(^5\)Side trivia: can you find the hidden easter egg and identify the fictional author of this software?
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 must have their accounts created by the admin user before being able to use the system. 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.

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: regular users versus users with administrator permission (simply called “administrators” hereafter). The special local “admin” user has permanent implicit administrator permission.

1.11. Administrators can grant or revoke administrator permission 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: The administrator Administrators 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. The administrator Administrators 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:** The administrator Administrators shall be able to add, modify, and remove assets within the system; users shall be able to review them.

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).

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.\(^6\)

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,

\(^6\)Removed 2020-02-10: This requirement has been withdrawn.
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”).

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\(^7\): 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).

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.

2.2.2. The administrator seeking to remove an asset will be prompted with a confirmation dialog first.

2.2.3. Users may view a table of stored assets showing all the short-form fields.

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.\(^8\)

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 3 (page 11) for an example of one possible implementation of such a feature. It is possible to combine this with req 2.2.4.1.

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 the asset is connected to managed PDUs. Given that this has to query PDU Networx 98 Pro, 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.

\(^7\)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”.

\(^8\)Removed 2020-02-10: This requirement has been withdrawn.
2.3. **Rack Datacenter management**: The administrator Administrators shall be able to add, modify, and remove datacenters and racks within the system; users shall be able to review them.

2.3.1. **Racks** 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. The administrator 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).

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 2, page 10).

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.\(^9\)

3. Bulk import/export facility

3.1. The administrator Administrators 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.

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.

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

---

\(^9\)Removed 2020-02-10: This requirement has been withdrawn.
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.

6. **Power management**: Administrators and the user listed as an asset’s owner may power a piece of equipment on and off using network managed PDUs, provided the asset 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*; see definition 8).

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. 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*, 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. All users 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.
Figure 2: A example rack elevation made compactly printable.
Figure 3: 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.