CINMan

Computing Infrastructure Management System

Abinash Patra, Ajaykrishnan Jayagopal, Ajmeera Balaji Naik, S Sai Teja Reddy, Harshal Gawai

OUTLINE:

- Project Description
- Requirements

    a.Functional Requirements

    b.Non Functional Requirements

- Z Specification
- Use Cases
- Usecase Diagram
- Future Scope
- Test Cases
- Database Design
- Table Diagram
- Requirements Specifications in UML

PROJECT DESCRIPTION

Large organisations having several computers connected over a network often face difficulty in the day-to-day administration of the network. Since there is no centralized control available for handling the network, the system administrator spends a lot of time and resources in ensuring uniformity of hardware and software across the machines in the network. This results in loss of valuable man-hours which could otherwise have been redirected to other productive activities. CINMan aims to solve this problem by centralizing control of the entire network so that the system administrator can monitor and make network-wide changes from just one machine like his/her laptop or even a mobile device.

REQUIREMENTS SPECIFICATIONS

Following the guidelines and notations presented , the requirements for CINMan have been specified as detailed below.

FUNCTIONAL REQUIREMENTS

The main functional requirements of CINMan are the following:

- CINMan will facilitate the system administrator to view the list of active machines.
- CINMan will facilitate the system administrator to view additional details of specific machine(s) that include, current operating system(s) running on the machine, the hardware specifications that lay underneath.
- CINMan will facilitate the system administrator to monitor the status of the machines.
- CINMan will send regular reports containing summary of the user activity.It will also push notifications in the software.
- CINMan will facilitate the system administrator to access it without any platform constraints.

NON-FUNCTIONAL REQUIREMENTS

- CINMan will have a client-server model.
  - There will be a client daemon installed on each machine. This daemon runs in the background and continuously monitors the system logs and

other user activity and communicates with the server whenever necessary, or when the server queries for information via websockets.

- ➢ The server will receive data from the clients or may itself request specific +clients for data.
- ➢ The server receives a stream of data from the clients. The server needs to perform an analysis of the data and store only the relevant data and discard any useless or redundant data.
- ➢ Additionally, there will also be another web-server that hosts the Admin Client that is ultimately viewed by the user. The user can view the data in a nice format , perform actions or be alerted of events, by the admin client.

- ❖ The server will be coded in Python. The web server will use the Django framework.
- ❖ The daemon will also be written in Python.
- ❖ The Client will be coded in Javascript, HTML5 and CSS using the JQuery framework.

Z SPECIFICATION:

CINMan = UserLogin V

ViewMachines V

ViewMachineDetails V

ViewUsers V

ViewUserDetails V

ViewAlerts V

ViewLogs

The operation of CINMan will be in one of the ***modes*** specified above. Following is the description of the state variables and constraints of the software.

Type Definitions :

```
USER == string
```

```
NAME == string

PASSWORD == string
```

Input Variables :

```
Username? : USER

Password? : PASSWORD

IPAddress? : String
```

Output Variables :

```
//displays successful login message.

displayLogin! : String

//displays all the users.

displayUserList! : String

//displays all the machines.

displayMachineList! : String

//displays user details when a particular userid is clicked.

displayUserDetails! : String

//displays machine configurations like RAM, OS, Hard Disk and etc.

displayMachineConfigs! : String

// displays machine logs after a particular machine is selected.

displayMachineLogs! : String

// displays alerts that need to be taken care of, by the user.

displayAlerts! : String
```

State variables :

```
users :  USER [ ]

passwords : hash(USER, PASSWORD)

currentuser : USER
```

```
loggedIn : ( true, false )

noActionState : NOP

machines : MACHINE [ ]
```

Utility Functions:

```
//returns true if the ip button was clicked.

is_active(ip) : ( true, false) was_ip_clicked( ip ) : ( true, false)

//returns true if the view_machine button was clicked.

was_view_machines_clicked( button ) : ( true, false)

//returns true if the view_user button was clicked.

was_view_users_clicked ( button ) : (true, false)

//returns true if a particular user_id button was clicked.

was_user_id_clicked( USER ) : ( true, false)

//returns true if a view_activity button corresponding to a user was clicked.

was_view_activity_clicked ( MACHINE ) : (true, false)
```

User Login :

```
loggedIn = (username?, password?) ∈ passwords

currentUser = if loggedIn

                then username?

                else ""

displayLogin! = if loggedIn

                 then "Logged in"

                 else "Wrong username/password"
```

View Machines :

```
UserLogin

if was_view_machines_clicked:
```

```
        then displayMachineList!
    else noActionState
```

View Users :

```
    UserLogin
    if was_view_users_clicked:
        then displayUserList!
    else noActionState
```

View User Details:

```
    UserLogin
    if was_user_id_clicked( uid ) Λ uid ∈ users:
        then displayUserDetails!
    else noActionState
```

View Machine Details:

```
    UserLogin
    if was_ip_clicked( m ) Λ m ∈ machines:
        then displayMachineConfigs!
    else noActionState
```

View Logs:

```
    UserLogin
    if was_view_activity_clicked( ip ) Λ ip ∈ machines:
        then displayMachineLogs!
    else noActionState
```

View Alerts:

```
    UserLogin
```

```
displayAlerts!
```

USE CASES

Case1	View Machine List and Filter by Criteria
ID	**UC1**
Actor	System Administrator
Pre-conditions	1. The software should be installed on all machines in the lab.
Flow of events	1. On startup of the software, the server daemon queries all the clients for system information. The clients obtain the system specifications locally and send it to the server. 2. A list of IP addresses are displayed to the user along with an indication of whether the machine at IP is active or not. This is done by remembering which all IPs have historically been connected to the server. 3. On clicking any IP address, the use case **UC2** is triggered. 4. A filter toolbar is available at the top from which the user can select filters to filter the machine IP list to display only those machines that satisfy certain criteria. 5. The set of filters to choose from, are as follows: a. Presence or absence of specific software packages. b. Hardware specifications like RAM, Disk space. c. OS kernel or distribution.
Post-conditions	1. The list of filtered machines is displayed to the user on choosing appropriate filters.

Case2	View machine Specifications (H/W and S/W)
ID	**UC2**
Actor	System Administrator
Pre-conditions	1. The software should be installed on all machines in the lab.
Flow of events	1. On clicking any IP address, the use case UC2 a new window opens which displays the system specifications of the machine at that IP: a. Hardware Specifications i. Memory

	ii. Disk iii. Motherboard iv. Peripherals. b. Software Specifications i. OS Info 1. Kernel version 2. Distribution ii. Installed software list 2. Additionally, if the machine is currently active, then the list of currently logged in users is also displayed. 3. On clicking a username, the use case **UC5** is triggered.

Case3	Activity Alerts
ID	**UC3**
Actor	System Administrator
Pre-conditions	1. The software should be installed on all machines in the lab.
Flow of events	1. The administrator is shown a list of actionable alerts that have occurred since last login by the administrator. 2. Alerts will be raised for events like: a. Peripherals added or removed. b. Unauthorised actions like attempting to install new software packages, resetting the password. c. Hardware and software failures, eg. software incompatibility with kernel version. d. Multiple logins by same user from different IP addresses. 3. The details of the alert will be displayed along with the IP and MAC address of the machine from which it originated. In case of user actions, the details of the faulting user are also displayed. 4. The administrator can further investigate these alerts by viewing the logs of that particular machine (covered by **UC4**) 5. The administrator can choose to save specific alerts for future reference.
Post-conditions	1. The displayed alerts, on clicking, are deemed to be acknowledged by the user.

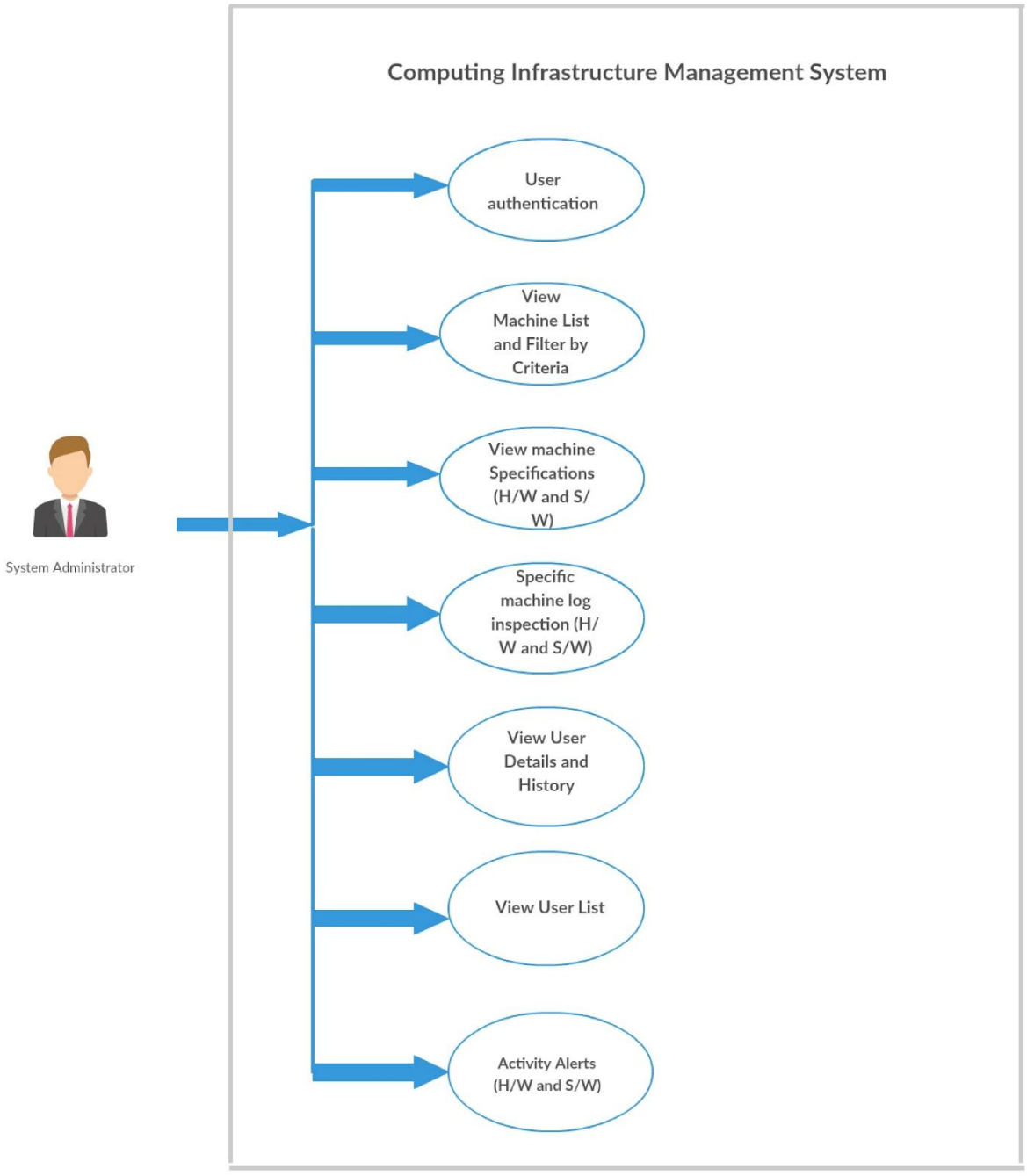
Case4	Specific machine log inspection (H/W and S/W)
ID	**UC4**
Actor	System Administrator
Pre-conditions	1. The software should be installed on all machines in the lab.
Flow of events	1. In **UC2**,there is a button called view log which will redirect to here(**UC4**). 2. The System administrator has to choose type of log he/she wish to inspect. 3. Request to the server is made and server responds with log information and will be displayed to the System Administrator.
Post-conditions	1. The specified log files are stored to the local system if requested by the administrator.

Case5	View User Details and History
ID	**UC5**
Actor	System Administrator
Pre-conditions	1. The software should be installed on all machines in the lab.
Flow of events	1. This use case is triggered by navigating to the user details page from either the user list page or from the machine details page. 2. The details of the user like username, email address, and previous usage history is displayed. 3. Usage statistics like data downloaded from the internet, softwares installed locally, list of IPs from which the user logs in. are also displayed. 4. The list of IPs in the lab into which the user is logged in is also displayed.

Case6	View User List
ID	**UC6**
Actor	System Administrator
Pre-conditions	1. The software should be installed on all machines in the lab.
Flow of events	1. The user can navigate to here from the main dashboard. 2. Displays the list of users having accounts in the lab along with whether or not they are currently logged in. 3. On clicking on any username, the use case **UC5** is triggered.

Case7	User Authentication
ID	**UC7**
Actor	System Administrator
Pre-conditions	1. The user should have valid credentials.
Flow of events	1. On opening the application, the user is first prompted to login with his/her credentials - a username and a password. 2. Only authenticated users are permitted to move past this step. 3. After successful authentication, the user can access the rest of the application.
Post-conditions	1. The user is logged in successfully and can access the rest of the application.
Secondary Scenarios	1. User credentials are incorrect. 2. User has forgotten password.

USE CASE DIAGRAM



TEST CASES

Initial State:

Users = {Ajay, Abinash, Saiteja, Balaji, Harshal }

Passwords = { (Ajay, ”33ajay33”), (Abinash, ”32abi32”), (Saiteja, ”51teja51”), (Balaji, ”34balaji34”), (Harshal, ”43gawai43”) }

loggedIn = false

LogIn:

Input	**Output**
username: “Balaji” password: “32abi32”	displayLogin = “Wrong username/password.” loggedIn = false currentUser = “”
username: “Saiteja” password: “51teja51”	currentUser = “Saiteja” loggedIn = true
username: “admin” password: “123admin”	displayLogin = “Wrong username/password.” loggedIn = false currentUser = “”

ViewMachines:

Input	**Output**
From any webpage in CINMan, if the **“Machines”** button in the navigation bar is clicked. ***OR*** A fresh valid login.	The list of machines in the network, in the given format. A colour code is also followed, as specified. **Format**: If the machine isn’t active: <machine_name><"("><ip_address><")"> If the machine is active: <machine_name><"("><ip_address><")">

ViewMachineDetails:

Input	Output
From the View Machines webpage in CINMan, if a particular machine is clicked.	A set of machine details are displayed, mostly, in tabular form. Hard disk usage and RAM usage are displayed as pie charts. • `IP Address` • `MAC Address` • `RAM Capacity (Pie Chart)` • `CPU Clock Frequency` • `Hard Disk Usage (Pie Chart)` • `OS Distribution` • `Kernel Version` • `Currently Logged in Users`

ViewUsers:

Input	Output
From any webpage in CINMan, if the **"Users"** button in the navigation bar is clicked.	The list of users in the network is displayed, in the given format. **Format**: `< username >`

ViewUserDetails:

Input	Output
From the View Users webpage in CINMan, if a particular user is clicked.	A list of **machines**, the user is *currently logged into*.

ViewMachineActivityLogs:

Input	Output
From the View Machine Details webpage in CINMan, if "**View Machine Logs"** is clicked.	A menu navigation bar is displayed with the following categories to select from. ● **Auth** -- Authentication Logs. ● **Peripherals** -- Peripheral Logs, e.g., if an external usb is connected. ● **Software** -- Software Logs. Upon selecting one of those categories, a bunch of logs are displayed with the corresponding timestamp.

ViewAlerts:

Input	Output
From any webpage in CINMan, if the **"Alerts"** button in the navigation bar is clicked. **OR**	The latest 20 alerts will be displayed.

Realtime ON/OFF Monitoring

Input	Output
Any of the machines goes offline/online.	If the machine goes offline, in the “**Machines”** webpage, that machine turns green to red within 60 seconds. I.e, <machine_name><"("><ip_address><")"> to <machine_name><"("><ip_address><")"> If the machine goes online, in the “**Machines”** webpage, that machine turns red to green within 60 seconds. I.e, <machine_name><"("><ip_address><")"> to <machine_name><"("><ip_address><")">

External USB Alerts

Input	Output
Any external USB is connected/disconnected	Alert will pop up.

The tables in our database along with their primary keys are as follows:

1. **Machine** : (ip_address)

```
ip_address : String

mac_address : String

address_width : Integer

ADDRESS_WIDTH_CHOICES = ( ( 1, '64-bit'), (2, '32-bit') )

ram_capacity : Integer

ram_description : String

cpu_speed : Float

cpu_description : String

harddisk_capacity : Float

harddisk_description : String

motherboard_description : String

os_distro : String

kernel_version : String

active : Boolean
```

2. **MachineUser** : (user)

```
name : String

user : String

last_logged_in_date : Time

last_logged_in_machine : String, fk

last_failed_login_date  : Time

failed_login_count : Integer

suspicious_activity_count : Integer

number_of_simultaneous_logins : Integer

currently_logged : Boolean
```

3. **MachineLoginSession** : (machine, user)

```
machine : String, fk

user : String, fk

login_time : Time

logout_time : Time

data_Downloaded : Integer

data_uploaded : Integer
```

4. **LogEntry** : (machine, timestamp, text)

```
machine : String, fk

timestamp : Time

log_entry_type :

TYPE_CHOICES = (  (1, "auth.log"),

                  (2, "kern.log"),

                  (3, "daemon.log"),

                  (4, "dpkg.log"),

                  (5, "boot.log"),

                  (8, "lastlog"),

                  (9, "wtmp"),

                  (10, "peripherals")

)

text : String

user : String

severity : String
```

5. **Alert** : log_entry
6. **Software**: (name, version)

```
machine : String, fk
software_type : Integer
sudo_needed : String
name : String
version : String
```

7. **SoftwareInstallation** : (software, machine)

```
machine : String, fk
software : String, fk
last_used_date : Time
last_user : String, fk
```

8. **Peripheral** : (machine, model)

```
machine : String, fk
model : String
type : String
description : String
```

9. **CINManUser**: (user)

```
user : String, fk
fullname : String
primary_mail : String
secondary_mail : String
mobile_number : String
```

DATABASE DESIGN

Some of the assumptions we have made while designing the database are as follows:

1. There are 2 distinct category of users who will interact with the system. One group is modelled by the table **MachineUser** and represents the users that use the network machines on a day-to-day basis. The other group is modelled by the table **CINManUser** and represents the system administrators who use CINMan to monitor and control the network.
2. A global catalogue of softwares across all machines in the network is available and is modelled by the table **Software**. Details regarding the software installations on each machine is stored in the **SoftwareInstallation** table.
3. Each machine may have 0 or more peripheral devices connected to it, each of which has a type and a model number.
4. The machine has a single motherboard with a single CPU and RAM chip. Also, information regarding the hard disk is not stored in the Peripheral table but in Machine itself. The machine has a single OS installed on it (or if it has multiple OSes, CINMan only recognises the currently active OS). The OS is assumed to be a distribution of Linux.
5. Details regarding the current and previous logged in sessions of the MachineUsers is stored in the **MachineLoginSession** table.
6. LogEntry stores a log entry of the machine. All log entries of the machine are converted to a standard form and stored in the table.
7. Alerts represent the events of interest to the CINManUser. They arise from problematic log entries.

With this in mind, the table diagram is as follows.

Entity-Relationship Diagram

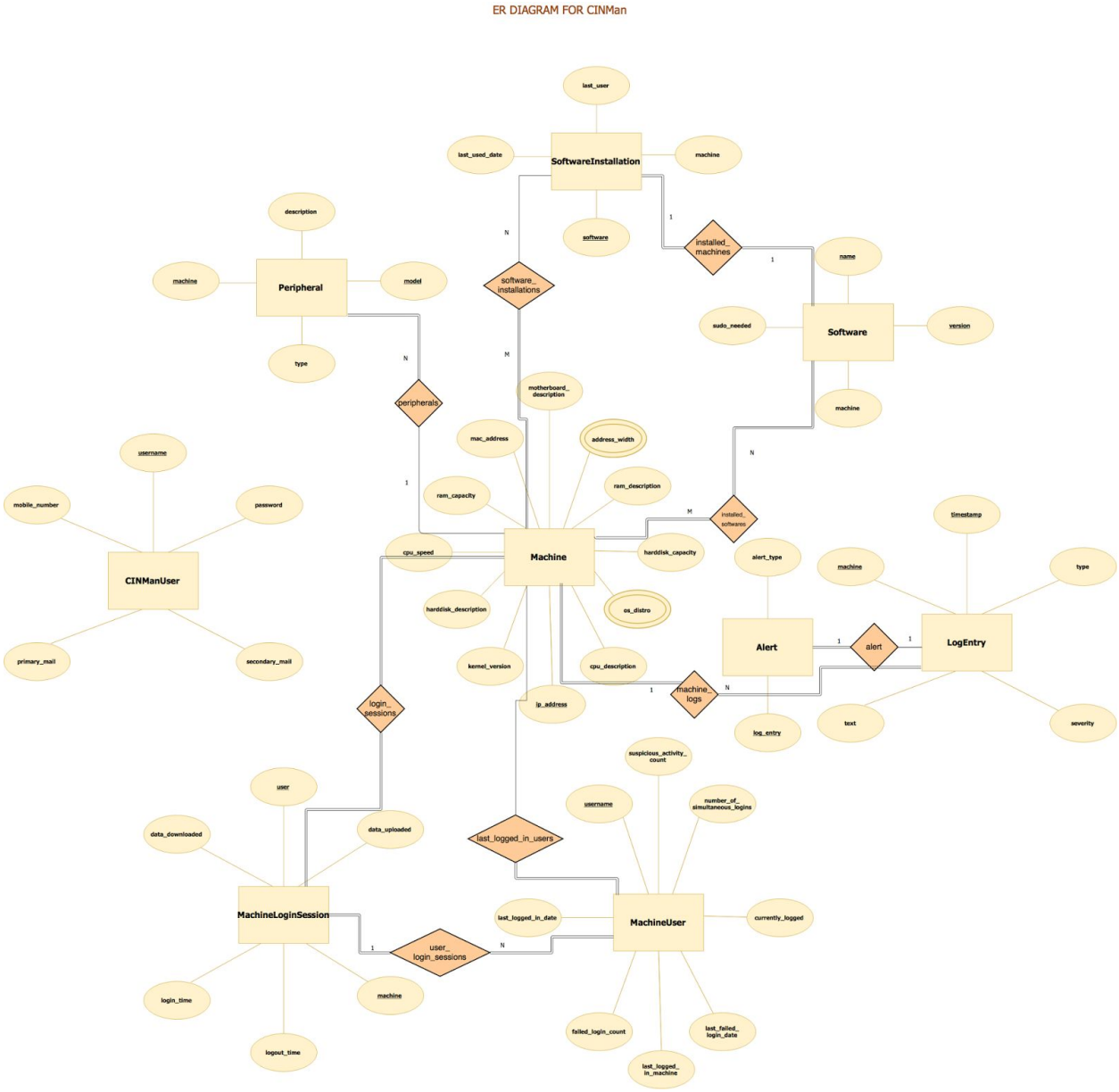
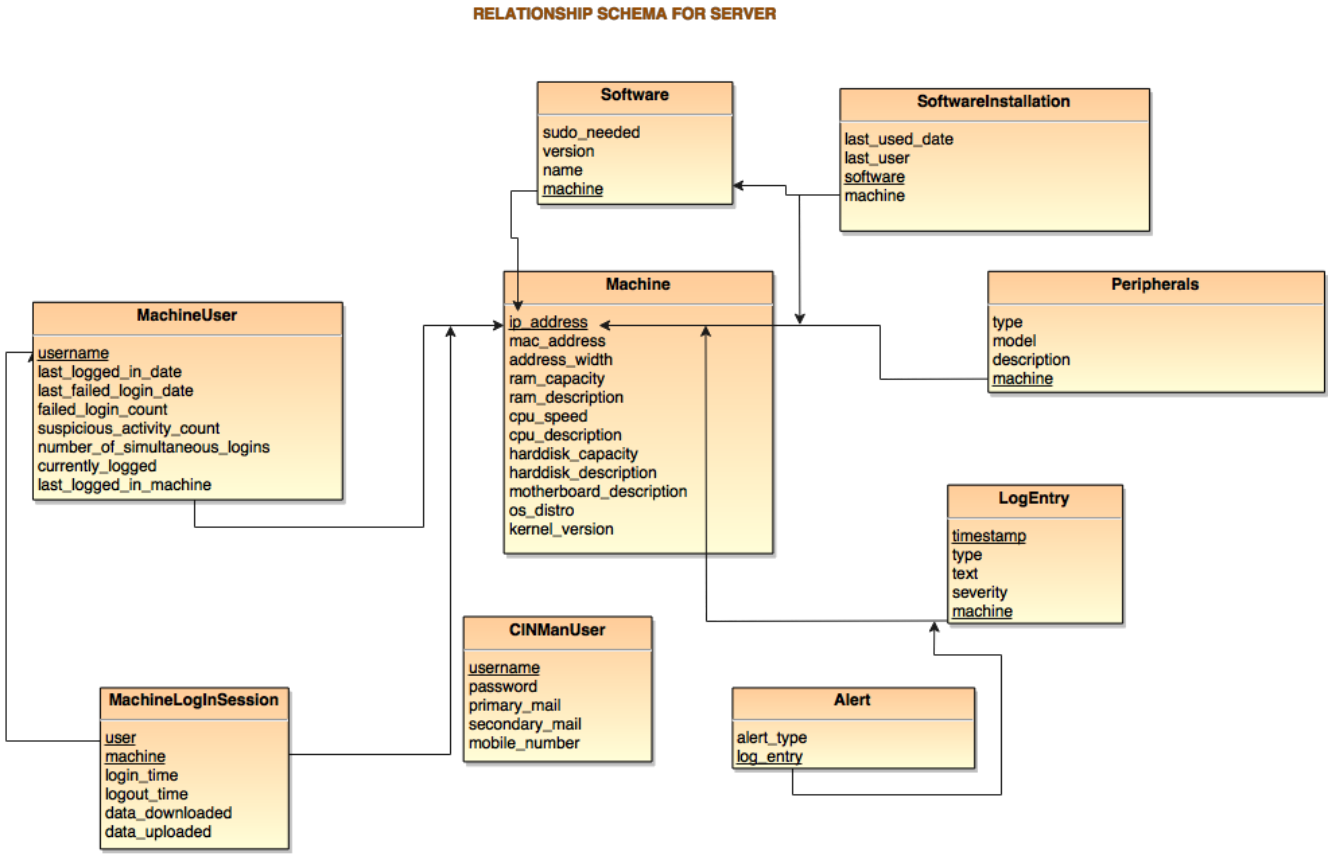


TABLE DIAGRAM

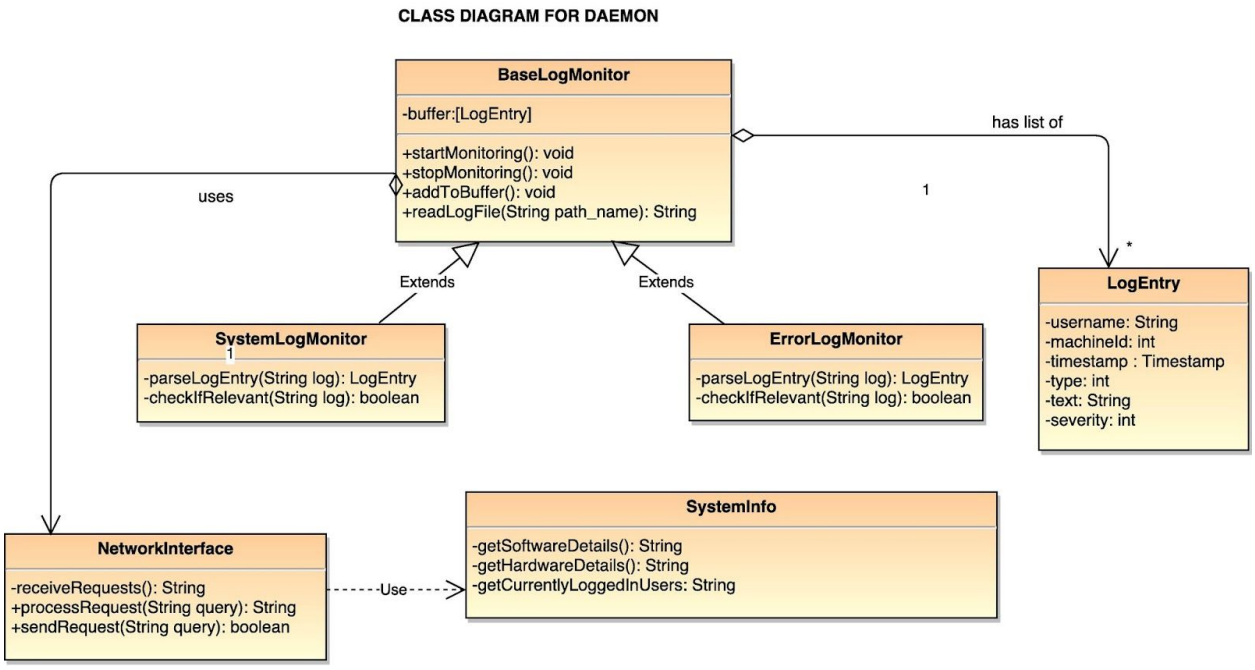


REQUIREMENTS SPECIFICATION IN UML (UNIFIED MODELLING LANGUAGE)

The requirements specifications as enumerated above are now presented in the form of UML diagrams.

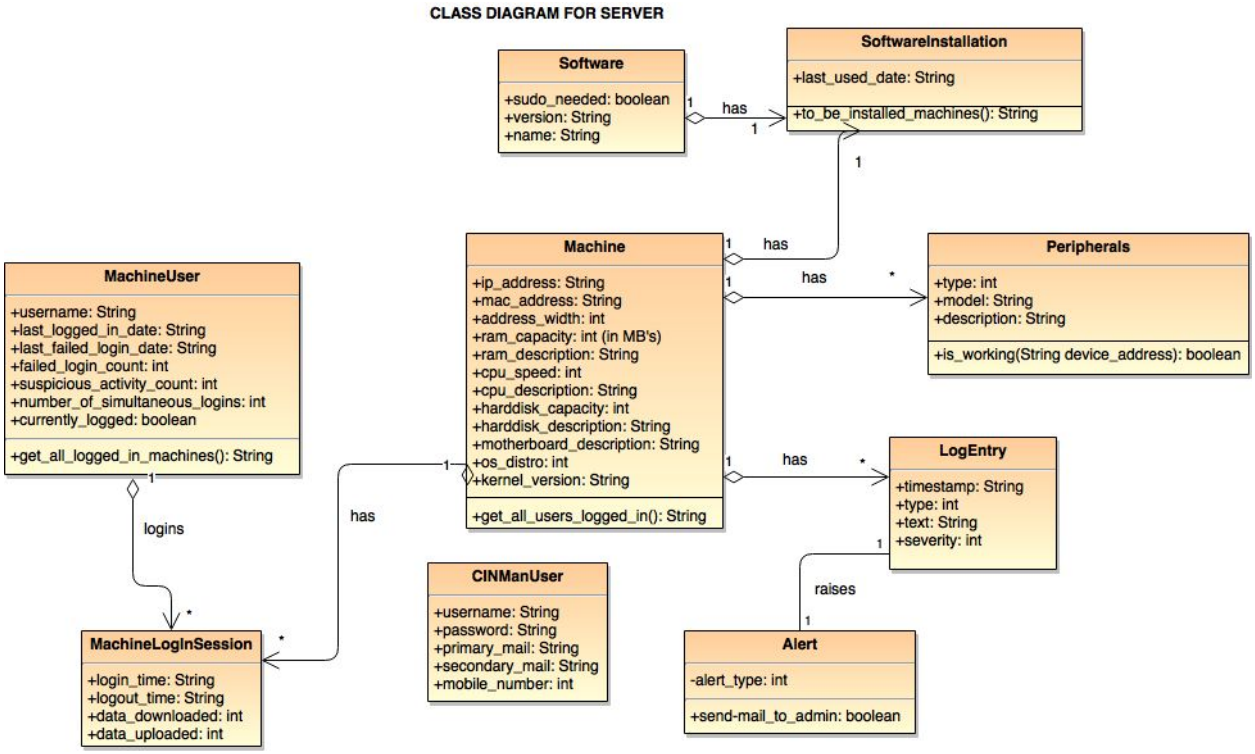
1. Class Diagram

    a. CINMan Machine Daemon



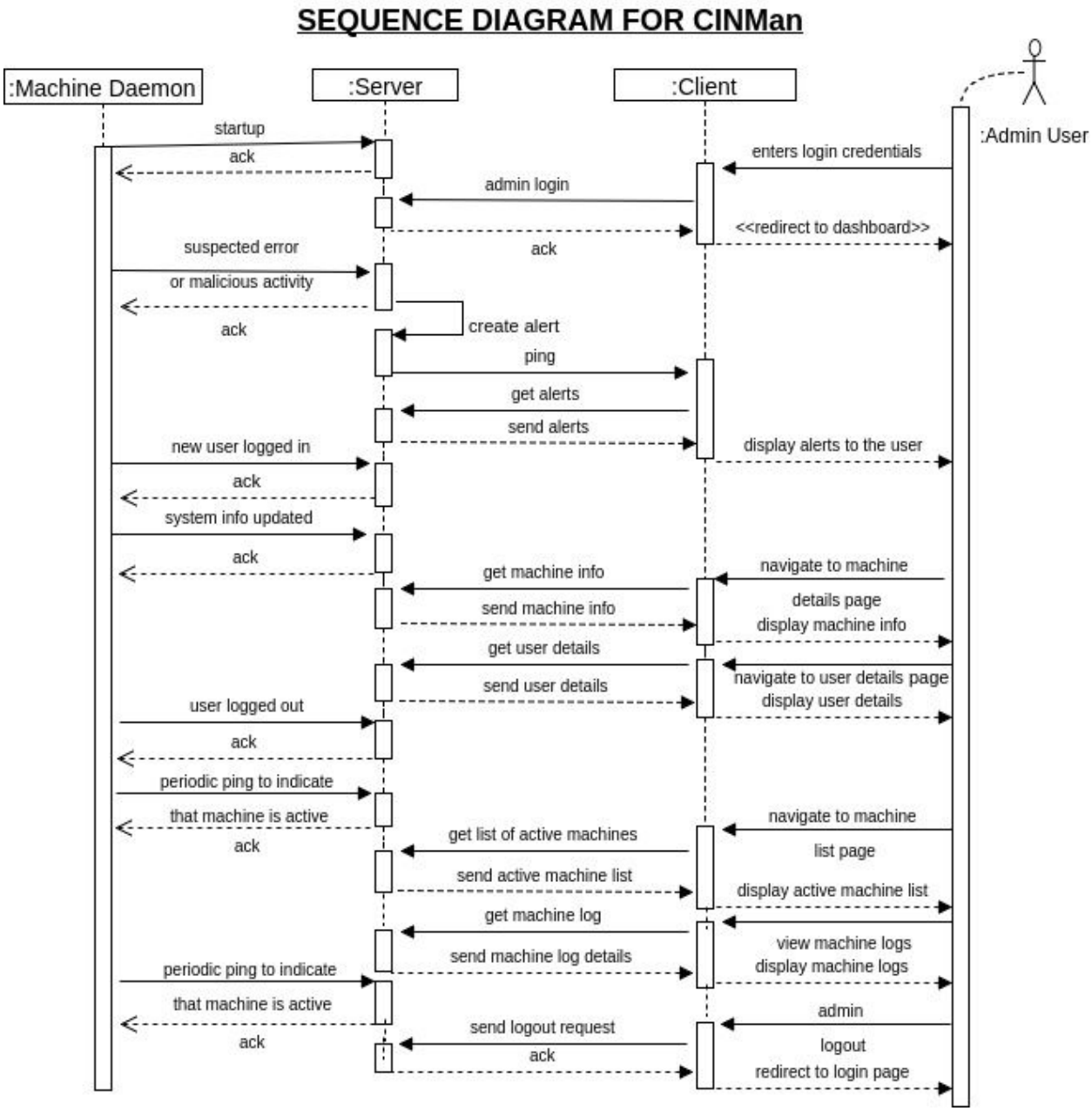
The MachineDaemon continuously runs in the background on each machine in the network and silently monitors all the system logs. The BaseLogMonitor class defines common functionality that can be used by the various other specific log monitor classes, for eg, SystemLogMonitor, ErrorLogMonitor which monitor the system-wide log and the error-log respectively. The BaseLogMonitor class maintains a buffer of LogEntry class objects which represent the entries of each log. The buffer has a size limit. Whenever the size of the buffer reaches this limit, then the daemon sends the LogEntrys to the server using the NetworkInterface class. Additionally, the NetworkInterface class can also receive requests from the server for SystemInfo. The SystemInfo class contains methods to obtain information regarding the machine on which it is running. This data is sent back to the server on being requested.

b. CINMan Intelligent Server



The CINMan Intelligent Server connects to the database and simultaneously manages connections to both MachineDaemons as well as WebClients. The server has classes corresponding to each of the tables in the database. This enables easy manipulation of data on the server. The process of connecting to the database and retrieving data from it are abstracted out by using suitable libraries. The server receives requests from the Client for data. The server processes these requests and responds with the required data. Similarly, the MachineDaemons supply the server with streams of log entries. The server processes these log entries and decides for which of them alerts should be raised. As soon as an alert is raised, the server sends

2. Sequence Diagram



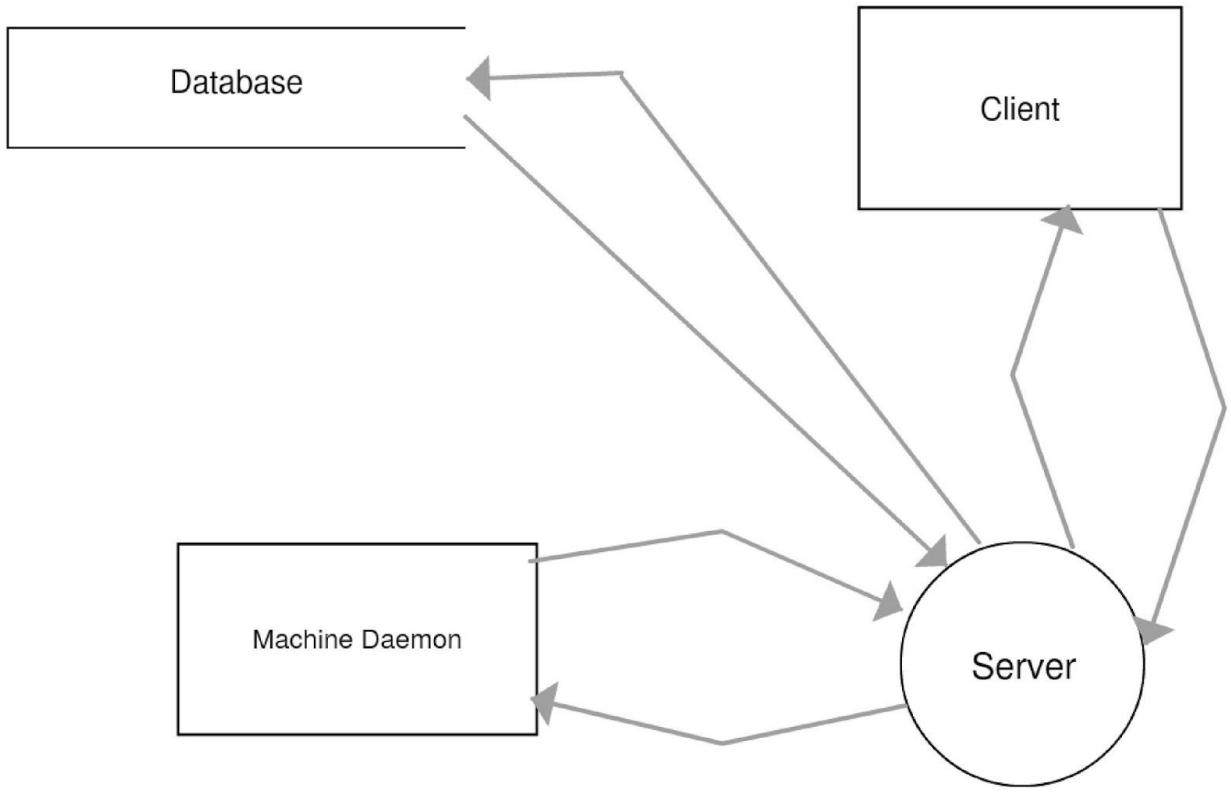
The sequence diagram displays the series of interactions between the different components of a software system beginning with the start-up of the software taking into account all possible user interactions with the system with time.

As we can see in the above diagram, the machine daemon is always active on the network machine and is continuously monitoring the logs for any activity. Whenever any machine starts, the machine daemon is also started on that machine and it sends a message to the central server

to inform it that the network machine is now active. The server adds this machine to the list of currently active machines.

The server is triggered whenever it receives a request from either the MachineDaemon or from the WebClient. Similarly, the client is triggered whenever the user interacts with it, and also if the server decides to initiate communication with the client for sending it alerts. The user may arbitrarily interact with the WebClient.

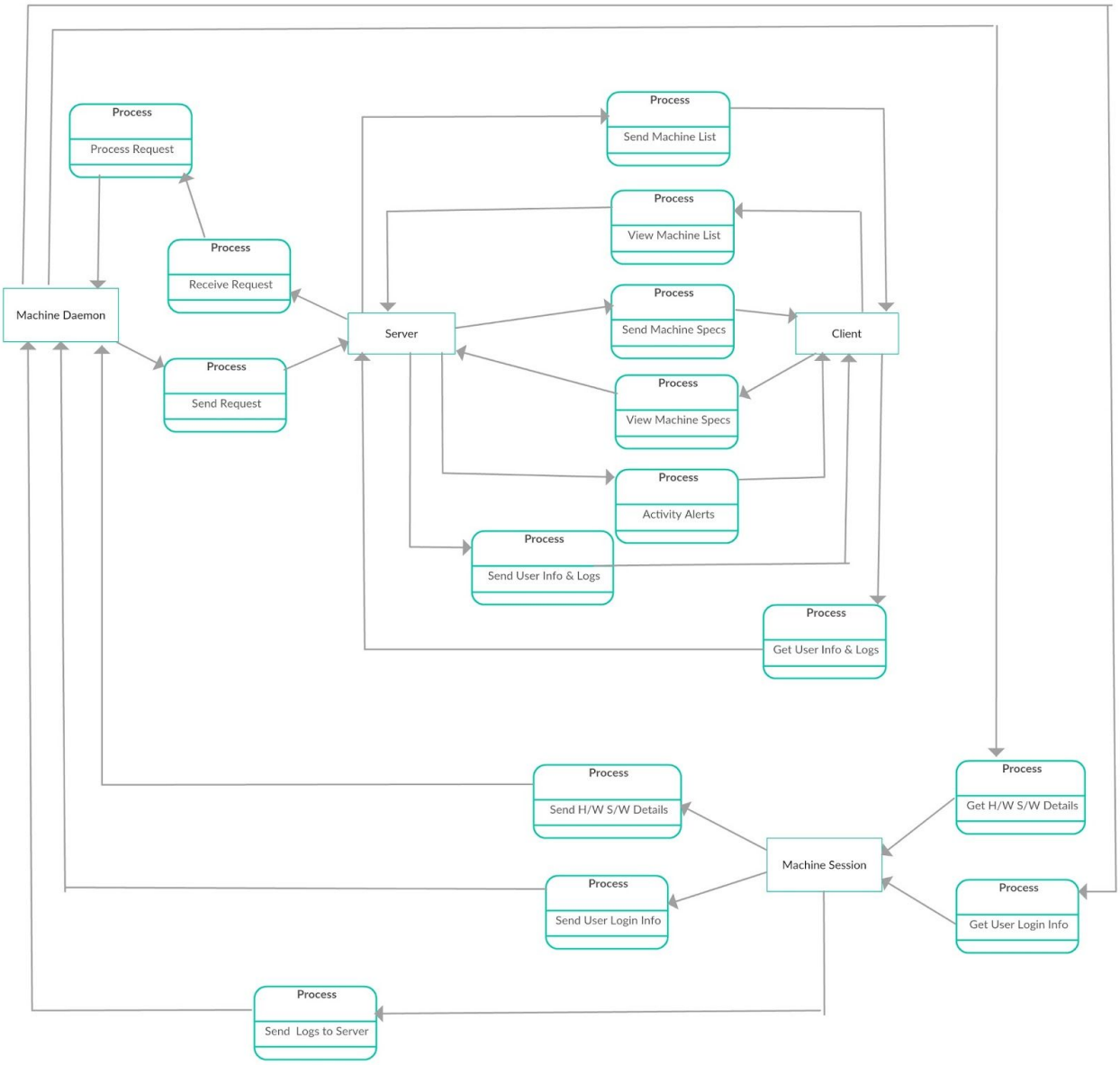
3. **Data Flow Diagram Level 0**



The Data Flow Diagrams describe the direction and flow of data between different modules of the system.The Data Flow Diagrams drawn here are of two types- Level 0 and Level 1.

The DFD-Level 0 diagram deals with the data flow on the top level of the system. The modules between which data flow happens are- Database,Client,Server and Machine Daemon. The machine daemon continuously sends log data to the server and the server keeps pinging the daemon for data. The server keeps sending data to database and retrieving data from it. The admin requests data from the server through the client.

Data Flow Diagram Level 1



The DFD-Level 1 diagram deals with data flow between various processes in the working program. This is a more elaborate description compared to DFD Level 0. It shows data flow between various processes comprising each module shown in DFD Level 0.

User Manual

Server:

sudo apt-get install python-pip

In server/CINMan/ :

pip install -r requirements.txt

./runner.sh

For shutting down the server:

./killer.sh

Daemon:

In daemon/ :

python install.py (only once, initially)

Enter server IP:port config, username, password

python daemon.py (subsequently)

Client:

Open client/login.html in a browser

FUTURE SCOPE

This project can be extended to support the following features:

1. Automatic installation of the client daemon on multiple machines.
2. The administrator can specify actions to be performed on individual or groups of machines. Eg. the administrator can specify a software list to be installed on a group of machines. CINMan will automatically install the correct software versions on the selected machines.
3. Automatic action on suspected malicious activity.
4. Configuration dashboard for the administrator to customize parameters for alerts and notifications.
5. Periodic mails shall be sent to the CINMan user informing him about specific alerts, and analysis.