Low Level Design: Web Portal

**Low Level Design**

**For**

**Web Portal**

Prepared by: UMT Dev Team Date: 14-08-2023

Reviewed by: Date:

Accepted by: Date:

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 14-08-2023 | 1.0 | First Release | UMT Dev Team |
| 11-Dec-2023 | 1.1 | Fine tuning with changes after development | UMT Dev Team |

**Table of Contents**

[1. Introduction 5](#_Toc165025625)

[1.1. Purpose 5](#_Toc165025626)

[2. LLD: Pending Approval 5](#_Toc165025627)

[2.1. View Task Detail 5](#_Toc165025628)

[2.2. Approve Task 6](#_Toc165025629)

[2.3. Reject Task 7](#_Toc165025630)

[3. LLD: Technician to Site Tagging Status 8](#_Toc165025631)

[3.1. Technician Listing 8](#_Toc165025632)

[a. input: user id of the supervisor who logged in 8](#_Toc165025633)

[3.2. Site Listing 8](#_Toc165025634)

[b. input: user\_id :Technician id of the selected user from the left panel 9](#_Toc165025635)

[3.3. Asset (Active/Passive) Viewing 9](#_Toc165025636)

[c. Screen Design 9](#_Toc165025637)

[d. input: asset\_site: Site id of the selected site. 10](#_Toc165025638)

[4. LLD: Operator Site View 10](#_Toc165025639)

[4.1. Operator Listing 10](#_Toc165025640)

[4.2. Site Listing 11](#_Toc165025641)

[4.3. Asset (Active/Passive) Viewing 12](#_Toc165025642)

[5. LLD: Site Asset View 12](#_Toc165025643)

[5.1. Site Listing 12](#_Toc165025644)

[5.2. Site Map View 13](#_Toc165025645)

[5.3. Asset (Active/Passive) Viewing 14](#_Toc165025646)

[5.4. View Details 15](#_Toc165025647)

[5.5. View Attributes 16](#_Toc165025648)

[5.6. View Asset History 17](#_Toc165025649)

[6. LLD: Audit Trail 18](#_Toc165025650)

[6.1. View Audit Trail 18](#_Toc165025651)

[6.2. Search 19](#_Toc165025652)

[7. LLD: Process Maker integration 20](#_Toc165025653)

[7.1. PROCESSMAKER Login 20](#_Toc165025654)

[7.2. Project Create 22](#_Toc165025655)

[7.3. Pending Approval 22](#_Toc165025656)

[7.4. assign task to PM users 24](#_Toc165025657)

# Introduction

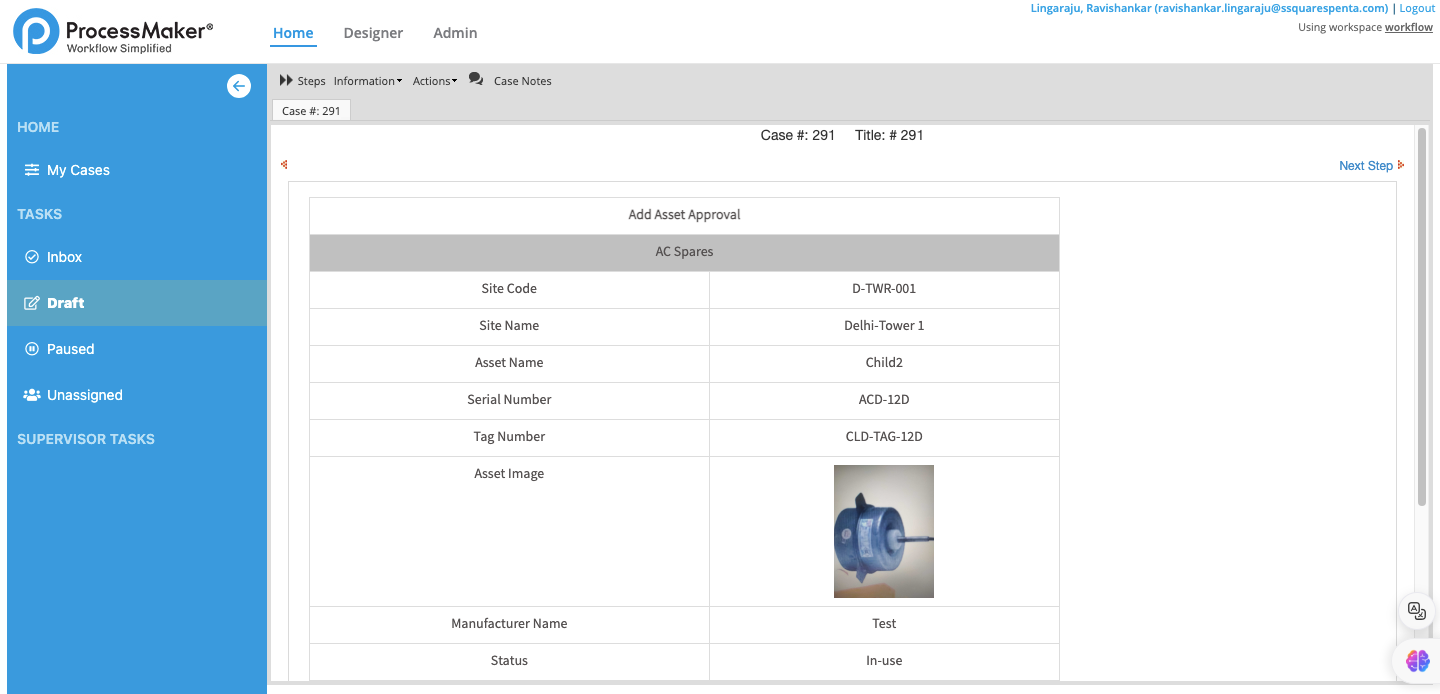
## Purpose

Developers refer to this document while developing the SATS web portal functionalities.

# LLD: Pending Approval

## View Task Detail

**Screen Design**



**input:**

* **API Endpoint** /get\_edit\_asset\_by\_process\_id
* **Method**: post
* **Request Body** {"project\_id":"72283700366192846388728064734461"}

**Processing**

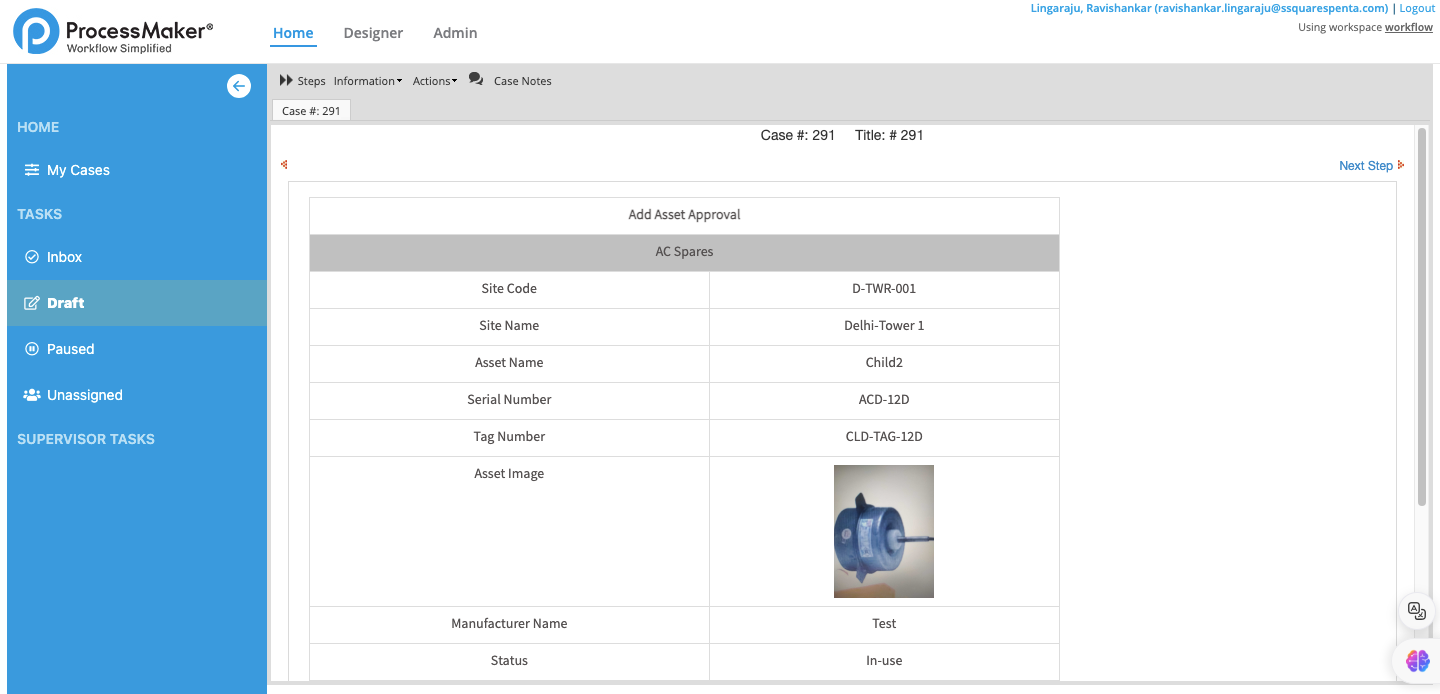
* User clicks the Pending Approval link and the pending approval list shows.
* Pending approval list is populated from the processmaker API (/api/1.0/workflow/cases/draft)
* Clicking on view details on any case opens (PM\_SERVER\_URL/sysworkflow/en/neoclassic/cases/opencase/{case number}") after login to processmaker using the token received from processmaker login
* To display the case details with asset data processmaker use a curl request from processmaker’s trigger with case id to SATS server’s API.
* Once the data is received from the API the data is displayed in a tabular format using html table.

**Output**

* Case details is displayed from the API response.

## Approve Task

**Screen Design**



**Input:**

* **project\_id:** Project id for the task

**Processing**

* User clicks the accept button from the task details from the processmaker.
* On button click it call the API like (SATS server URL/api/approve\_pm\_request\_asset)
* The api will update the **t\_pm\_approval,t\_asset,t\_far\_to\_ats**  table and the task is closed.
* The api will also insert the **mail\_log** table for sending mail through cron for approved tasks.

**Output**

* Task is approved and closed. The output of the

{

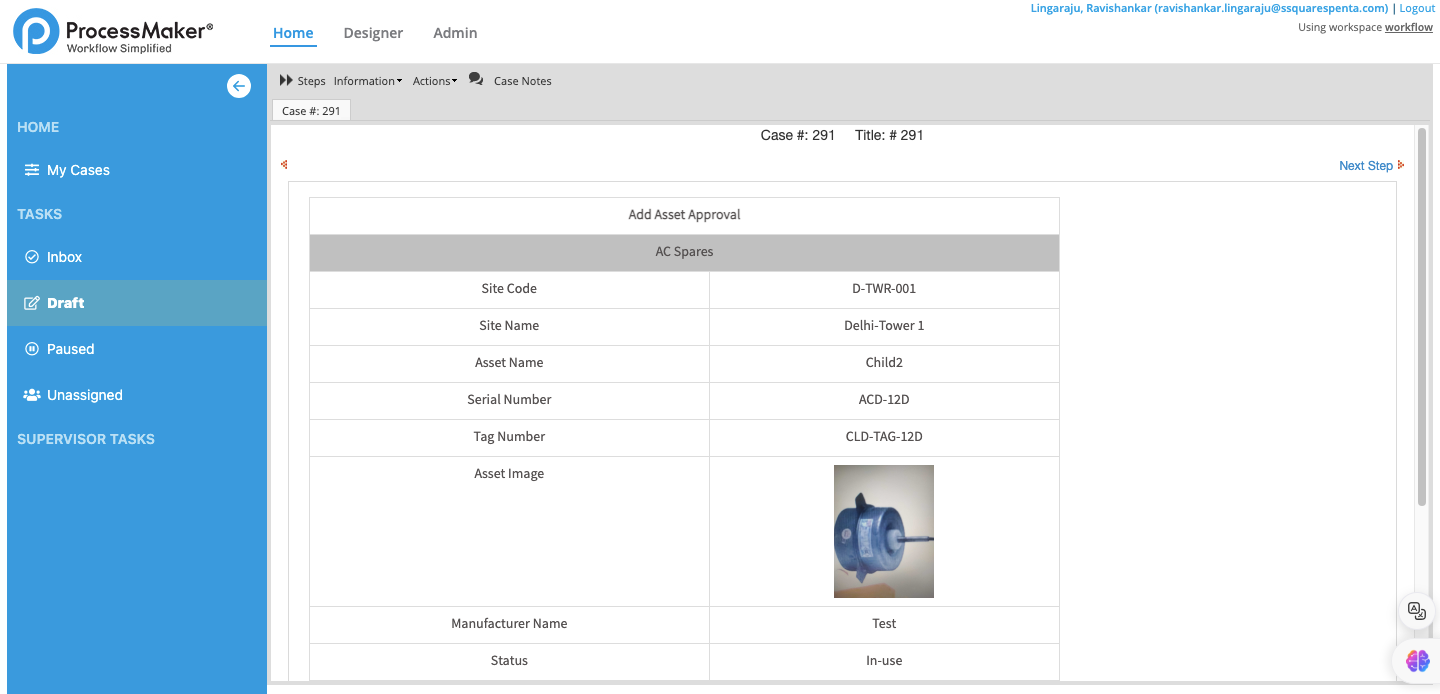
"status" => 200,

"message" => 'Assets Added Successfully',

}

## Reject Task

**Screen Design**



**Input:**

* **project\_id:** Project id for the task

**Processing**

* User clicks the reject button from the task details from the processmaker.
* On button click it call the API like (SATS server URL/api/reject\_pm\_request\_asset)
* The api will update the **t\_pm\_approval,t\_asset,t\_far\_to\_ats**  table and the task is closed.
* The api will also insert in the **mail\_log** table for sending mail through cron for rejected tasks.

**Output**

* Task is rejected and closed.

{

"status" => 200,

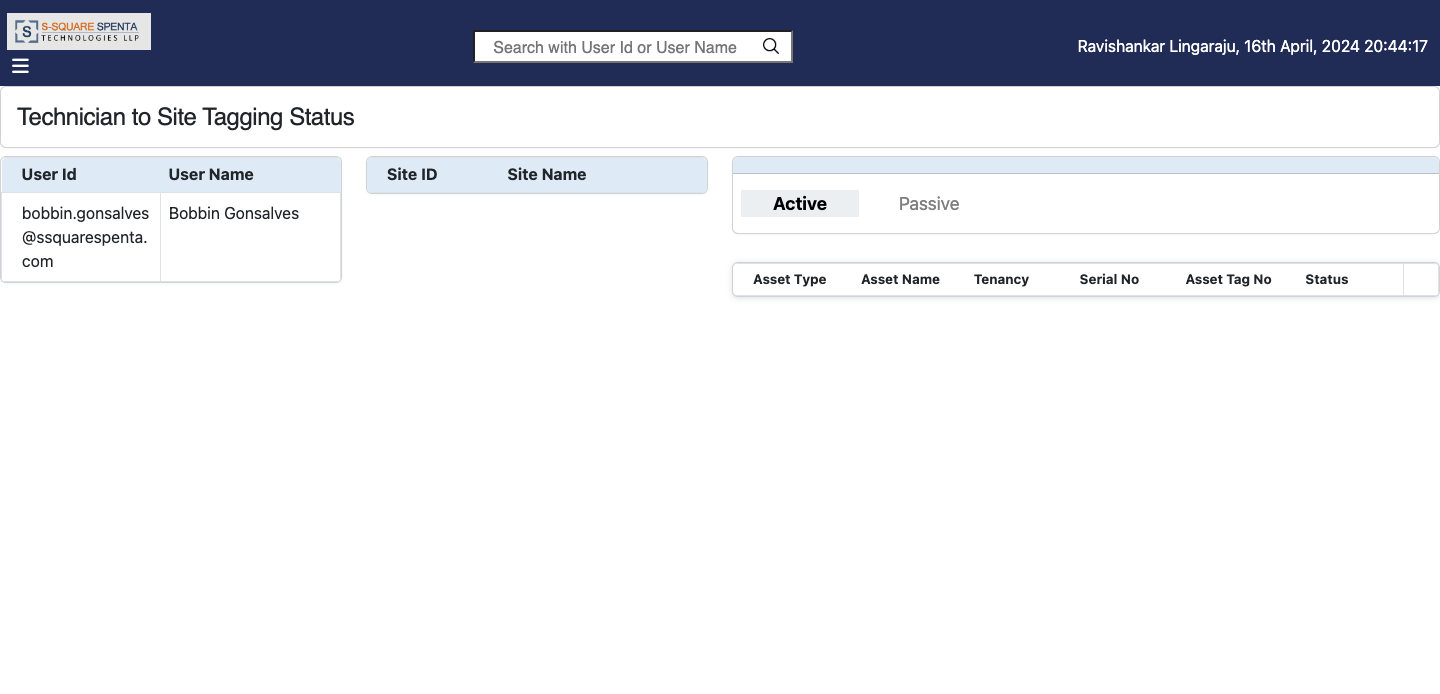
"message" => 'Asset Rejected Successfully',

}

# LLD: Technician to Site Tagging Status

## Technician Listing

**Screen Design**

****

## input: user id of the supervisor who logged in

**Processing**

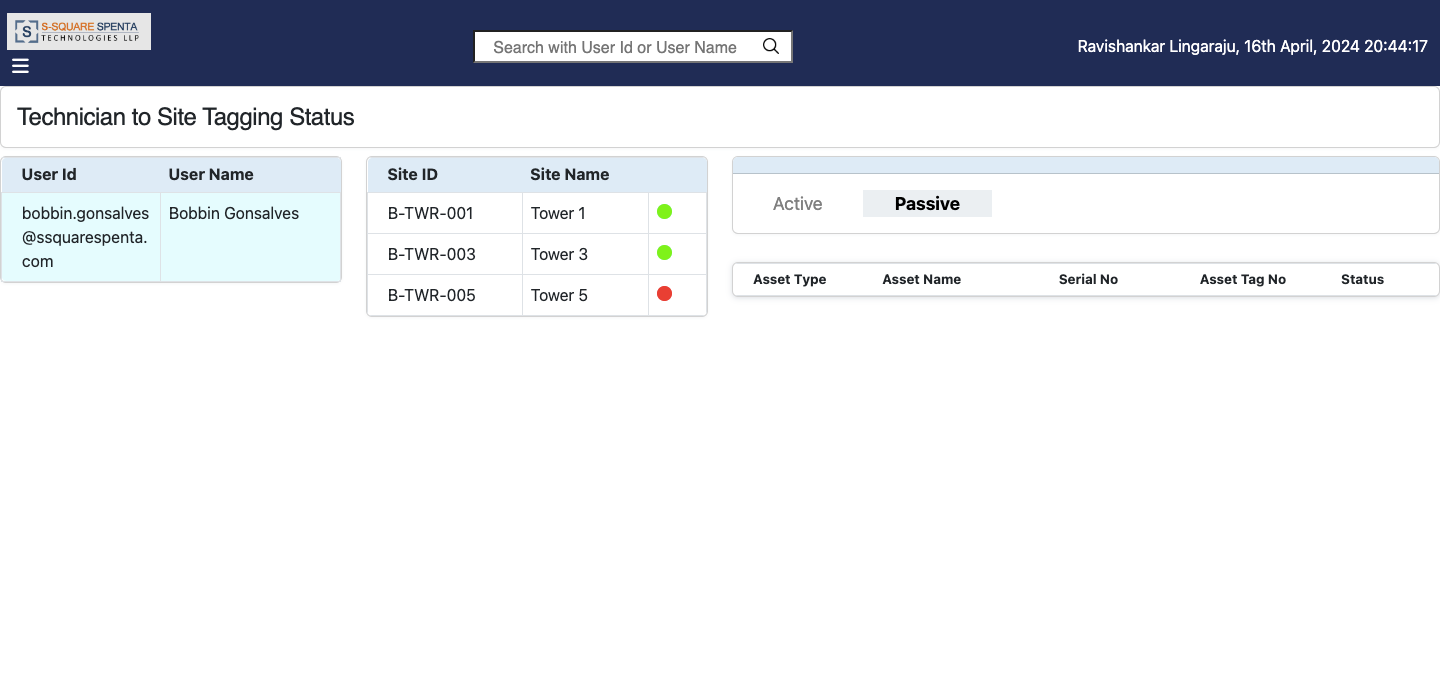
* User clicks the Technician to Site Tagging Status.
* List of technicians**(userid ,user name)** assignedunder the supervisor shows from the table **t\_technician\_supervisor\_mappinng**

**Output**

* Listing of the technicians assigned under the logged in supervisor.
* The **Auth::id()**  function of Laravel’s Auth gives the user id of the login user.
* From the **t\_technician\_supervisor\_mappinng** table the system matches the **supervisor\_id** to get the technicians listing.

## Site Listing

**Screen Design**

****

## input: user\_id :Technician id of the selected user from the left panel

AJAX get Request: **SATS URL**/technician\_site\_map\_jason?user\_id=77

**Processing**

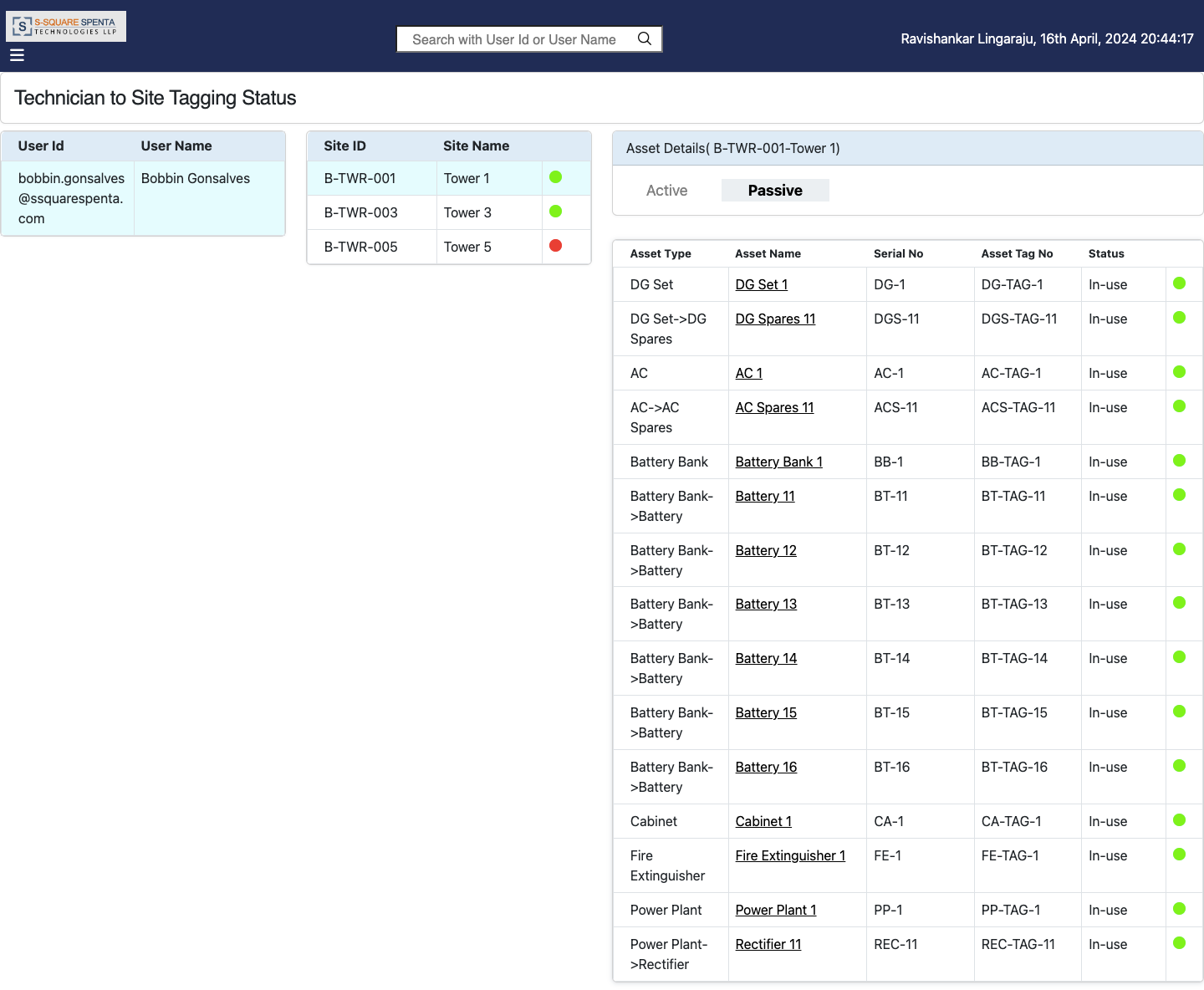
* From Technician to Site Tagging Status page shows list of technicians**(userid ,user name)** assignedunder the supervisor.
* clicking on a particular user will display the list of sites assigned to the user using ajax call with url (**SATS server URL//technician\_site\_map\_jason?user\_id={userID}**)
* The api check the t\_user\_location table for data.

**Output**

* The assigned site for any technician is listed from the table **t\_user\_location** table where 'ul\_user\_id = $request->user\_id join with **t\_location** table;

## Asset (Active/Passive) Viewing

## Screen Design

****

## input: asset\_site: Site id of the selected site.

**AJAX get Request**: SATS server URL//technician\_site\_active\_passive?asset\_site=243

**Processing**

* From site listing clicking on a particula site will call ajax to get all the active and passive assets for the task

**Output**

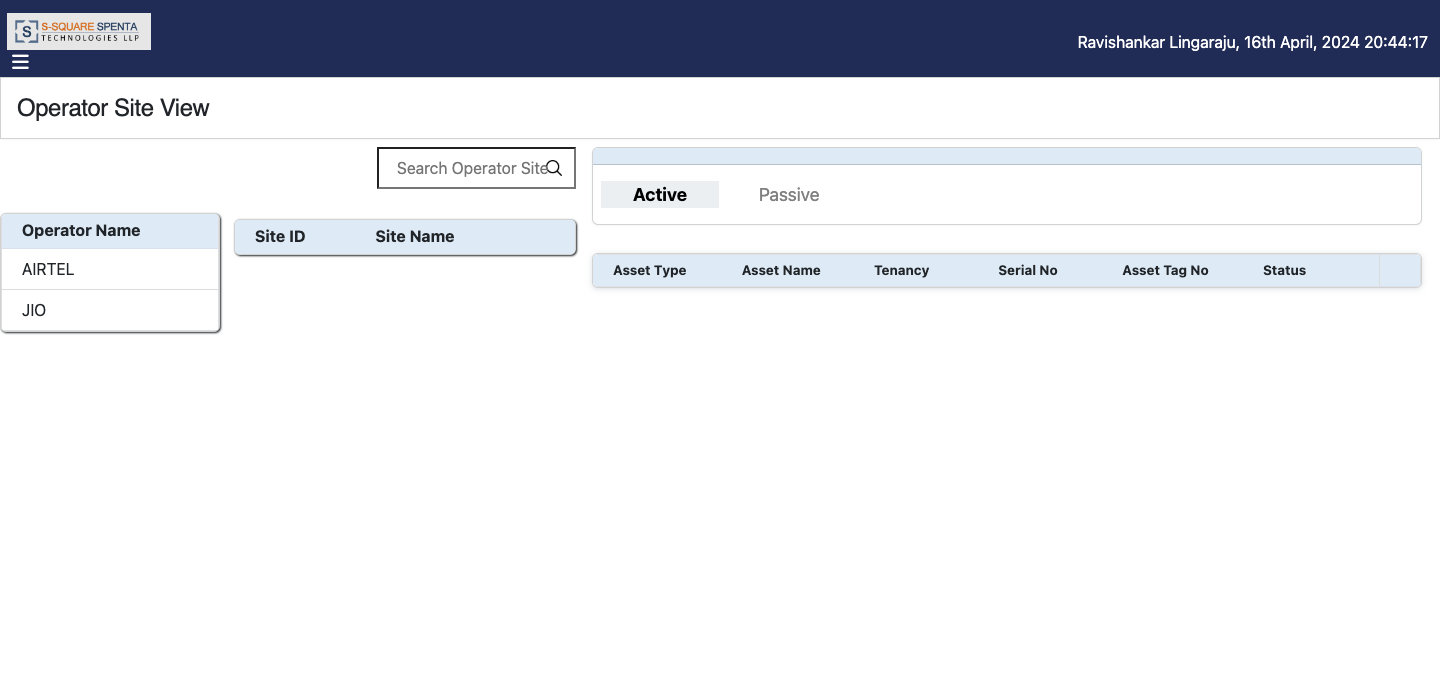
* The assets for the particular site are listed.

}

# LLD: Operator Site View

## Operator Listing

**Screen Design**

****

**input:** Login user id.

**Processing**

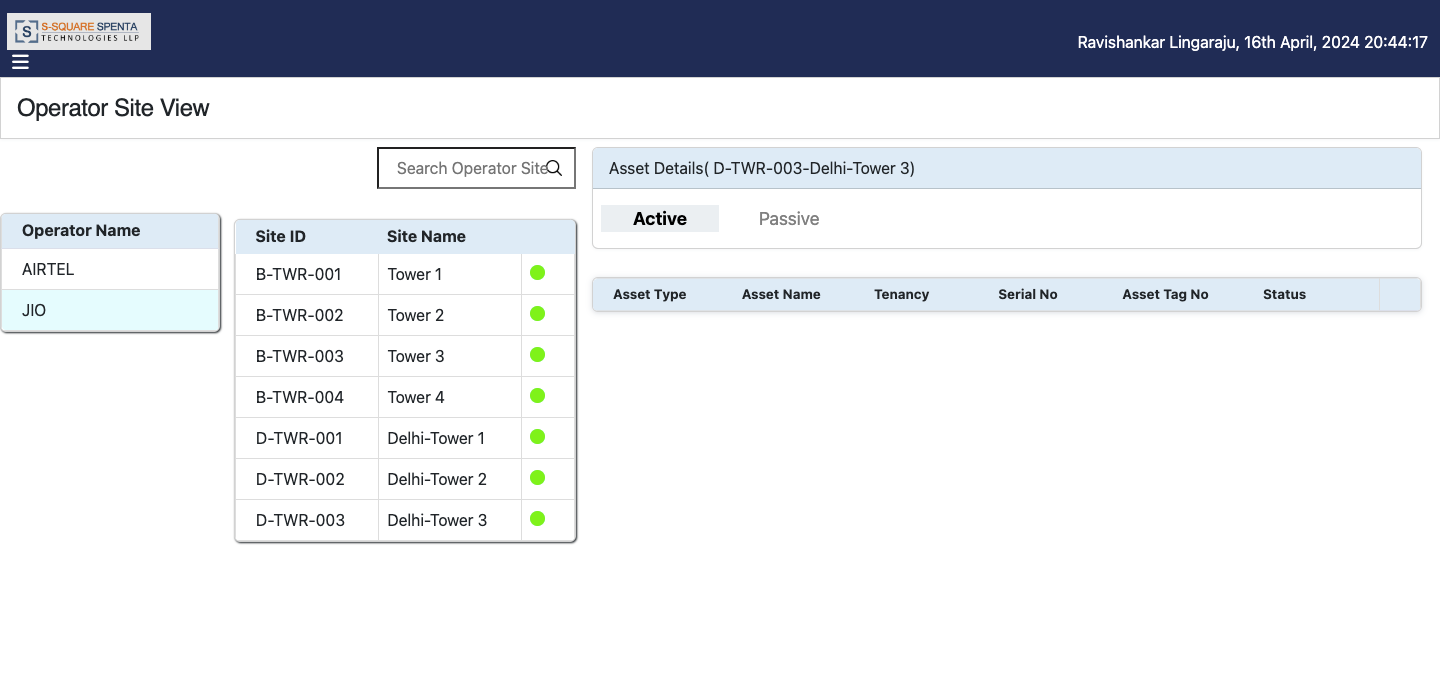
* User clicks Operator Site View from the left menu link.
* List of operators show sin the first panel from [**product**](http://115.113.197.12:85/public/adminer.php?pgsql=localhost&username=postgres&db=ats-testing&ns=product) **» t\_operator** table.

**Output**

* All operators are listed **t\_operator** table**.**

## Site Listing

**Screen Design**

****

**input: operator\_id:** id of the mobile operator**.**

**Ajax Request:** https://ats.esquaressquaredev.com/operator\_site\_jason?operator\_id=1

**Processing**

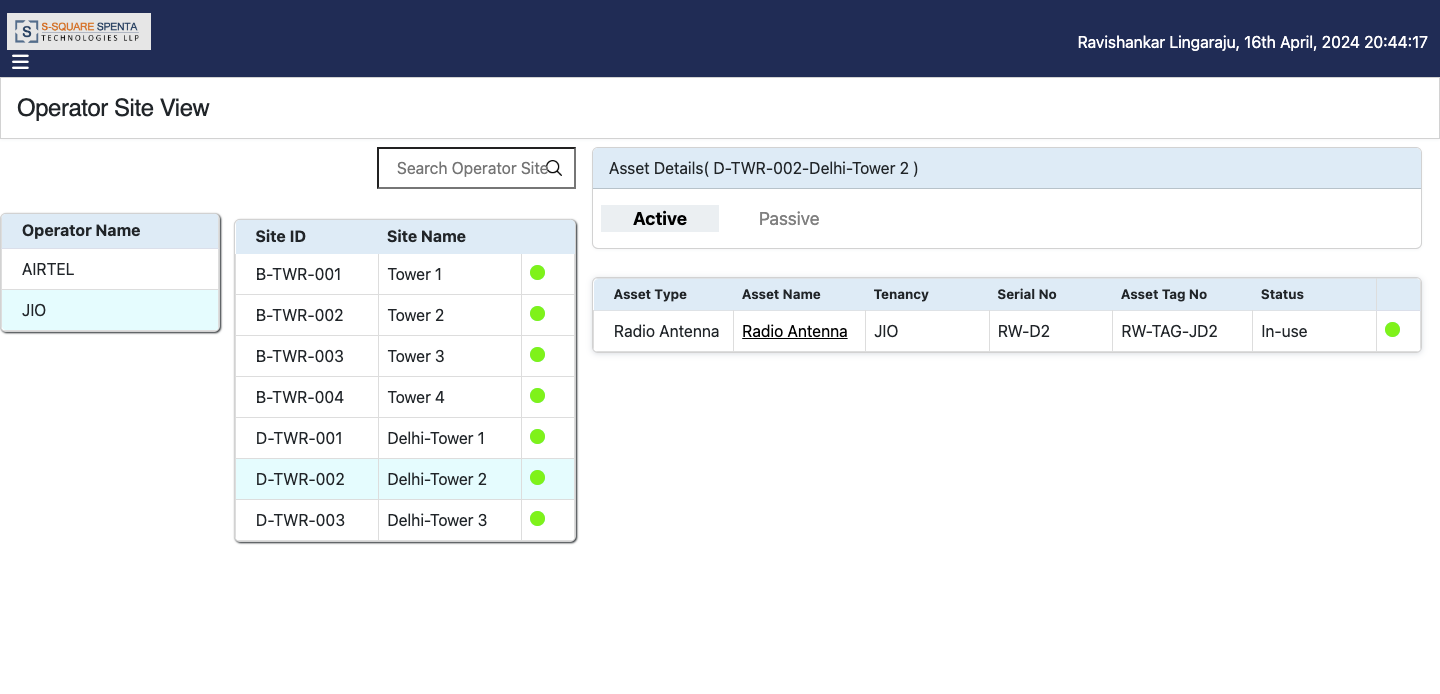
* User clicks Operator Site View from the left menu link.
* List of operators shows in the first panel. Clicking on any operator shows the list of sites in which the operators has active assets using ajax call passing the **operator\_id** as query parameter which checks in the **t\_asset** table for the **operator\_id**.

**Output**

* List all the sites where the active assets are present for an operator.

## Asset (Active/Passive) Viewing

**Screen Design**

**Input:**

**Ajax Request:** https://ats.esquaressquaredev.com/operator\_active\_passive?site\_id=243&operator\_id=1

**Processing**

* User clicks Operator Site View from the left menu link.
* List of operators shows in the first panel. Clicking on any operator shows the list of sites where operator has active assets. Clicking on any site shows all the active and passive assets using ajax call with the query parameters **site\_id** and **operator\_id.**

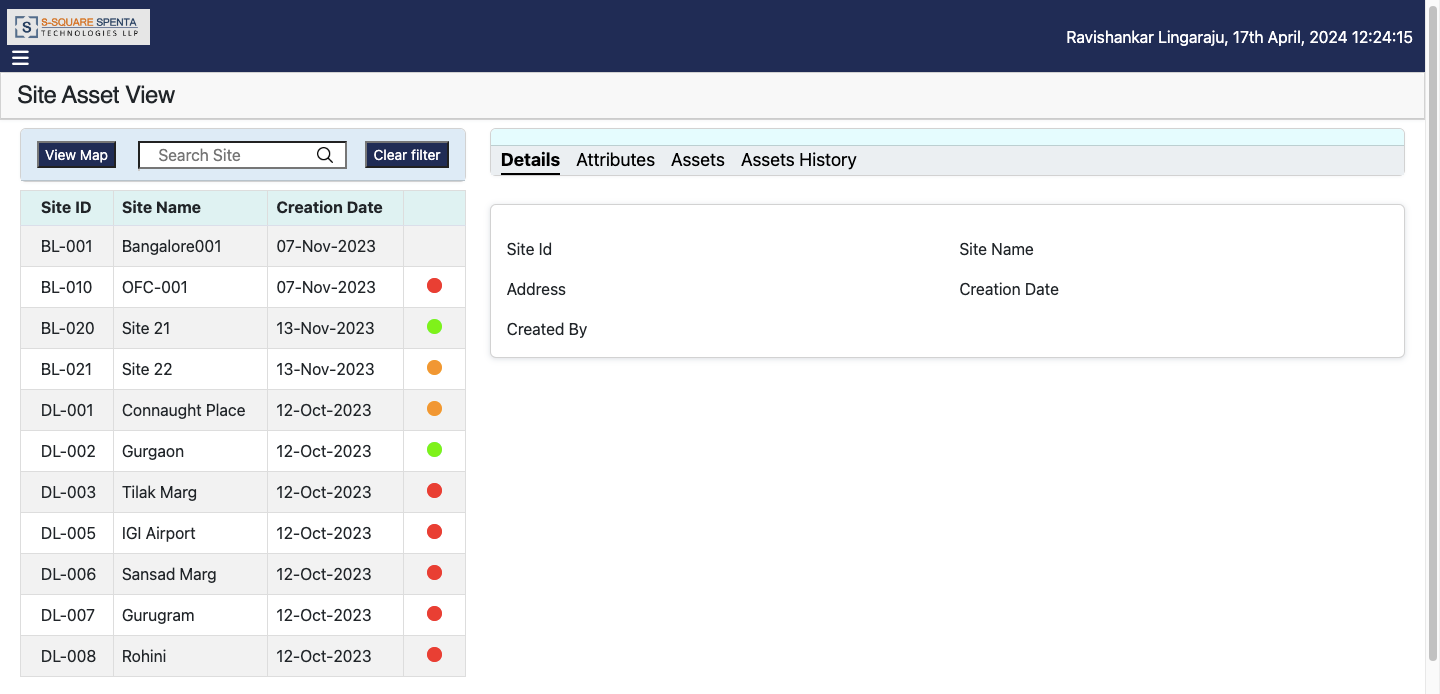
**Output**

* Listing of active and passive assets are displayed for any site.

# LLD: Site Asset View

## Site Listing

**Screen Design**

****

**Input:**

**User ID:** Login user’s id using **Auth::id()**

**Processing:**

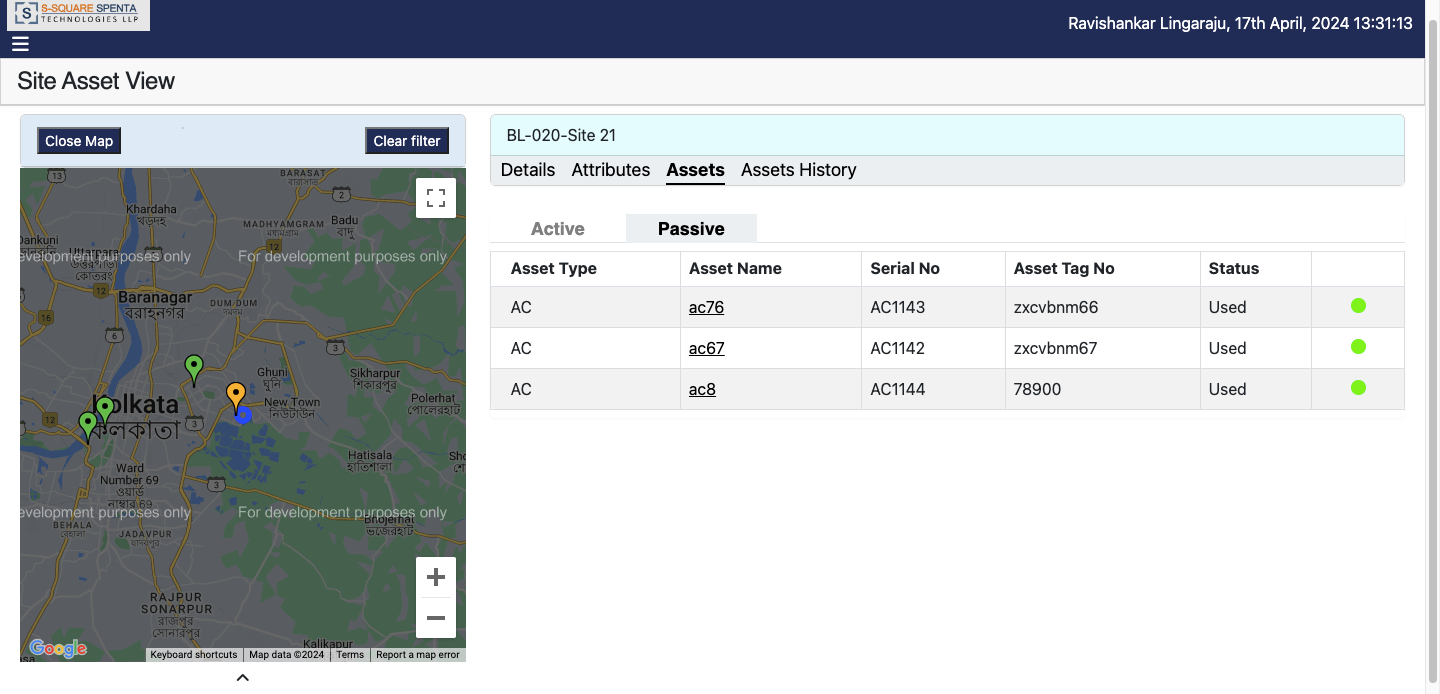
* User clicks **Site Asset View** from the left menu link.
* Listing of all sites from **t\_location** table is listed.
* Tagging status color code shows for the sites.
* User can search sites by typing on the search box. When typing a auto suggestion displayed using ajax call search\_location?search={search text}.
* Clicking the clear filter buton clears all the filter and lists all the sites.

**Output**

* System checks **t\_technician\_supervisor\_mappinng** for the technicians under the logged in supervisor id.
* Then system get all the **location id** from **t\_user\_location**
* Then the system searches **t\_location** for the location ids.

## Site Map View

**Screen Design**

****

**Input**

**User ID:** Login user’s id using **Auth::id()**

**Processing**

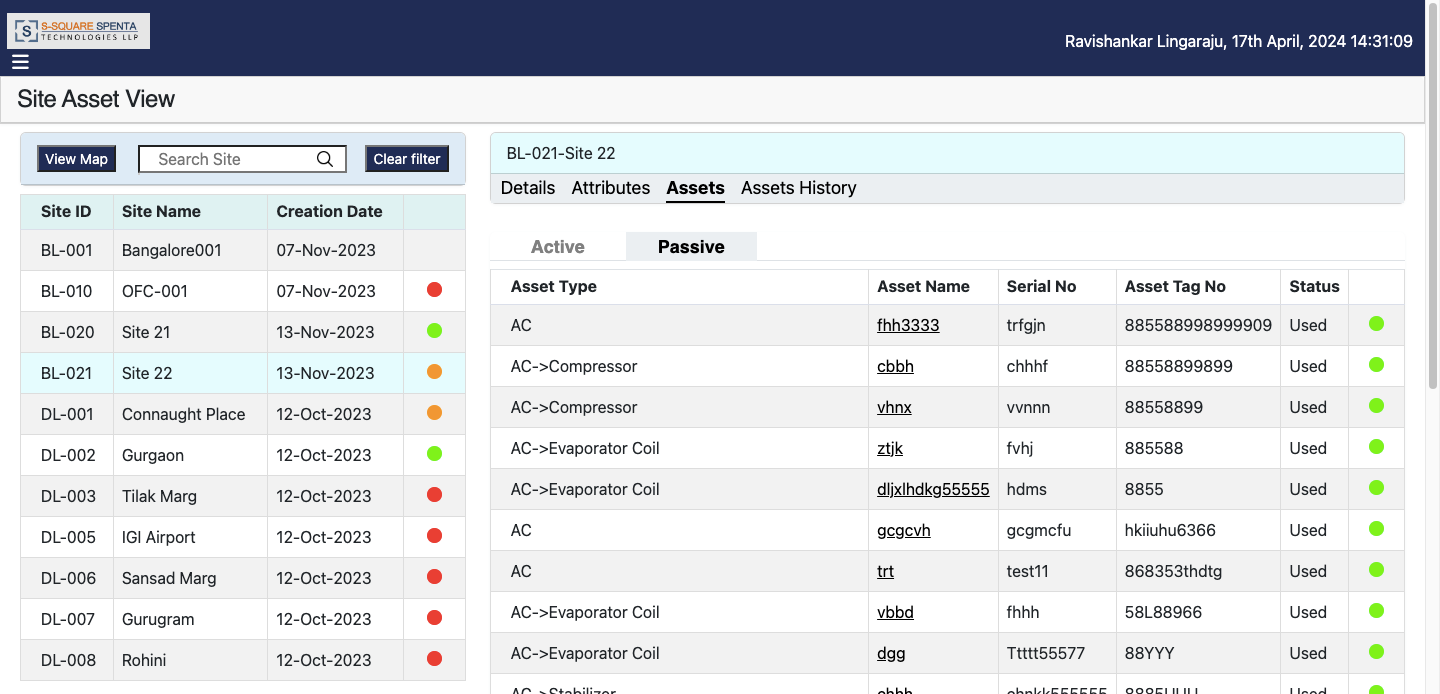
* User clicks **Site Asset View** from the left menu link.
* From site asset view page user clicks **View Map** button and the button changes to **Close Map**.
* Google map opens with all the sites with markers and current location is shown as circle.
* The google map is integrated using javascript.
* clicking on any site opens a popup with assets with tagging status.
* clicking on the popup opens the site details in the right screen.

**Output**

* System checks **t\_technician\_supervisor\_mappinng** for the technicians under the logged in supervisor id.
* Then system get all the **location id** from **t\_user\_location**
* Then the system searches **t\_location** for the location ids.
* Using javascript for google map the markes are placed using current location .

## Asset (Active/Passive) Viewing

**Screen Design**

****

**Input**

**AJAX Request:** http://ats.esquaressquaredev.com/locationdb?location\_id=266

**Processing**

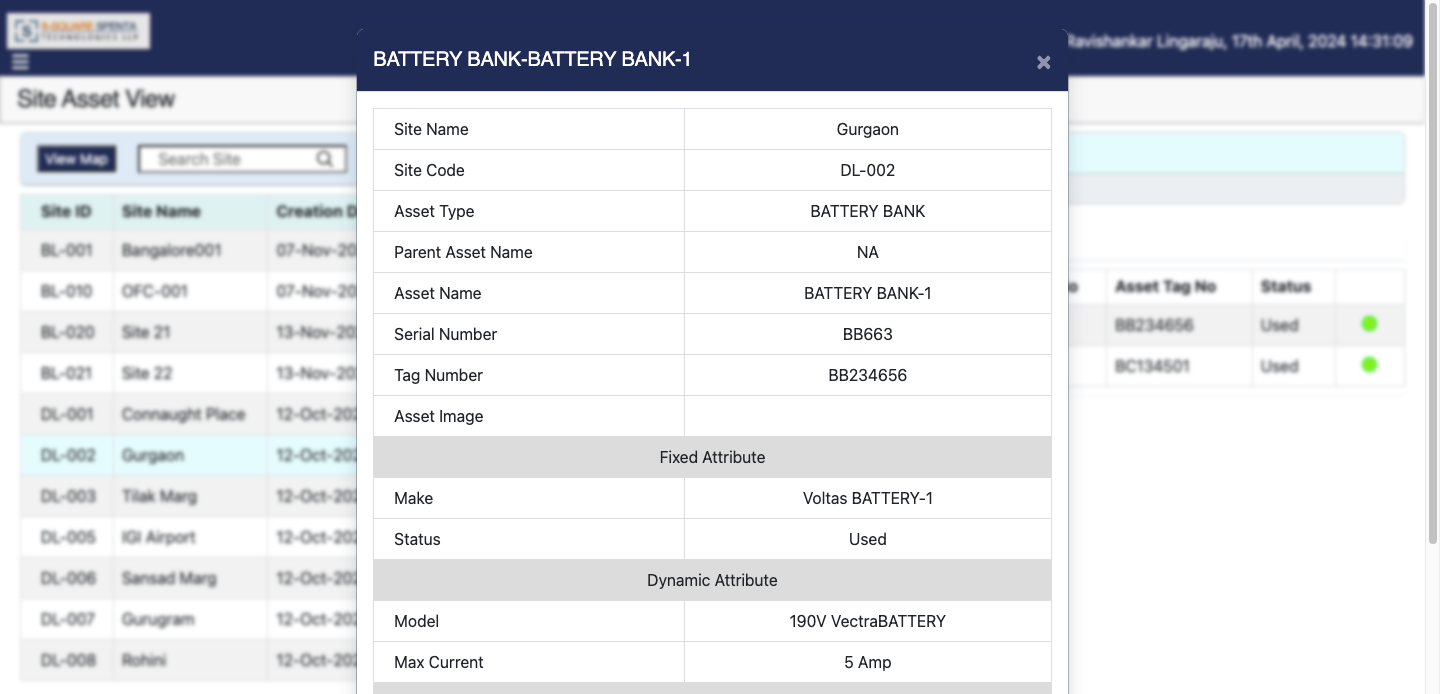
* User clicks **Site Asset View** from the left menu link.
* Listing of all sites from **t\_location** table is listed.
* Clicking on any site calls ajax with **location\_id** parameter fetch all assets from the **t\_assets** table.

**Output**

* Listing of all assets from **t\_assets**.

## View Details

**Screen Design**

****

**Input:**

**Ajax Request:** http://ats.esquaressquaredev.com/SingAsstDetails?asst\_id=307

**Processing**

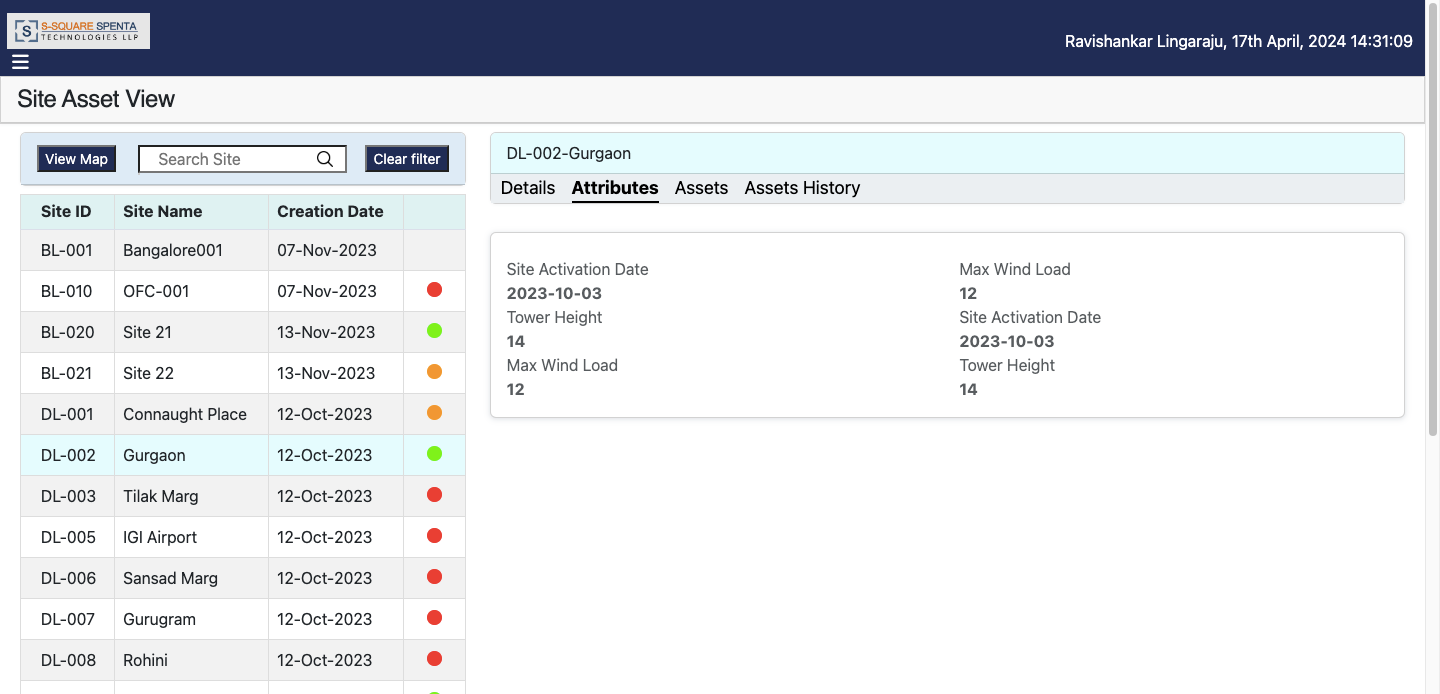
* User clicks **Site Asset View** from the left menu link.
* Listing of all sites from **t\_location** table is listed.
* Clicking on any site calls ajax with **location\_id** parameter fetch all assets from the **t\_assets** table.
* clicking on any asset one popup opens for the asset using ajax call from **t\_asset** table with **t\_asset\_attribute**

**Output**

* Assets details are shown from **t\_asset** table with **t\_asset\_attribute.**

## View Attributes

**Screen Design**

****

**Input:**

**Ajax Request:** http://ats.esquaressquaredev.com/locationdb?location\_id=242

**Processing**

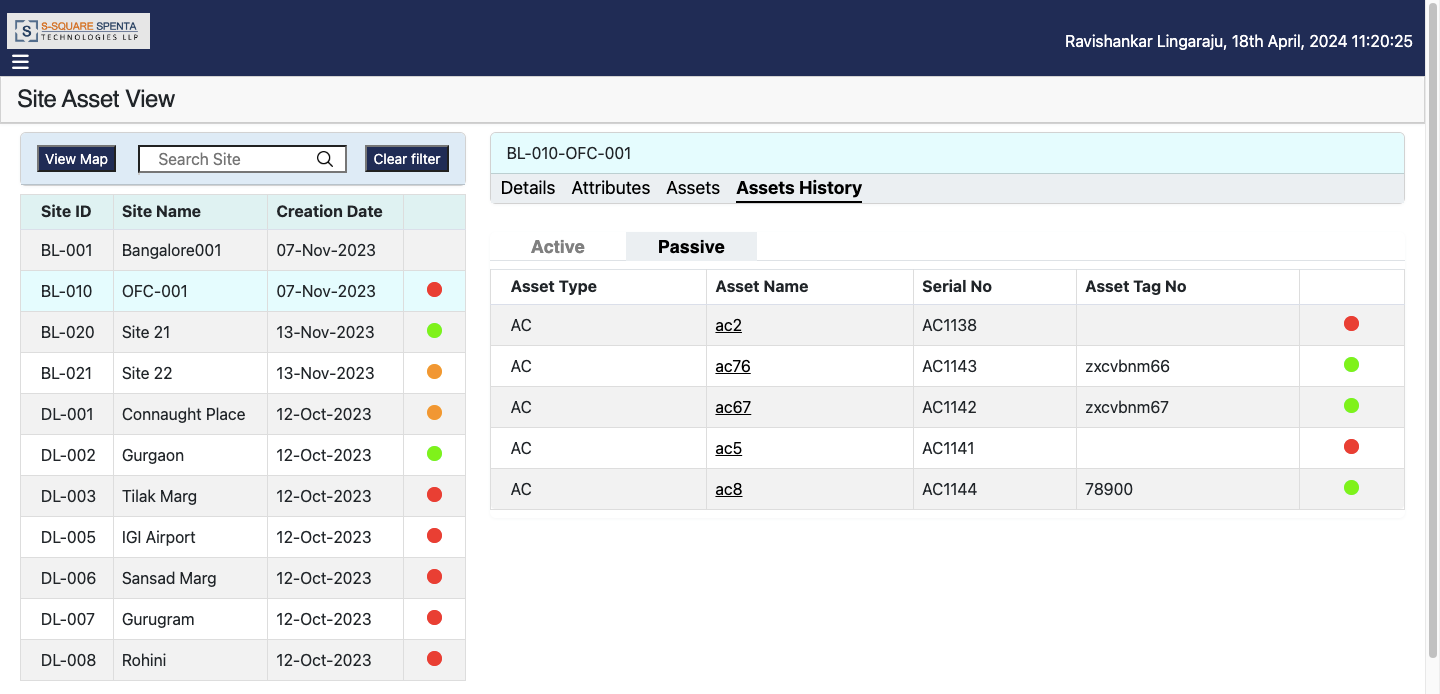
* User clicks **Site Asset View** from the left menu link.
* Listing of all sites from **t\_location** table is listed.
* Clicking on any site shows attributes link. Clicking on the link shows the attributes for that site from **t\_location\_attribute** table.
* .

**Output**

* Attributes details are shown from **t\_location\_attribute** table**.**

## View Asset History

**Screen Design**

****

**Input**

**Ajax request:** http://ats.esquaressquaredev.com/locationdb?location\_id=266

**Processing**

* User clicks **Site Asset View** from the left menu link.
* Listing of all sites from **t\_location** table is listed.
* Clicking on any site shows asset history link clicking on the link shows both active and passive assets which were present in the site.
* It searches **t\_asset\_history** table for move out date.

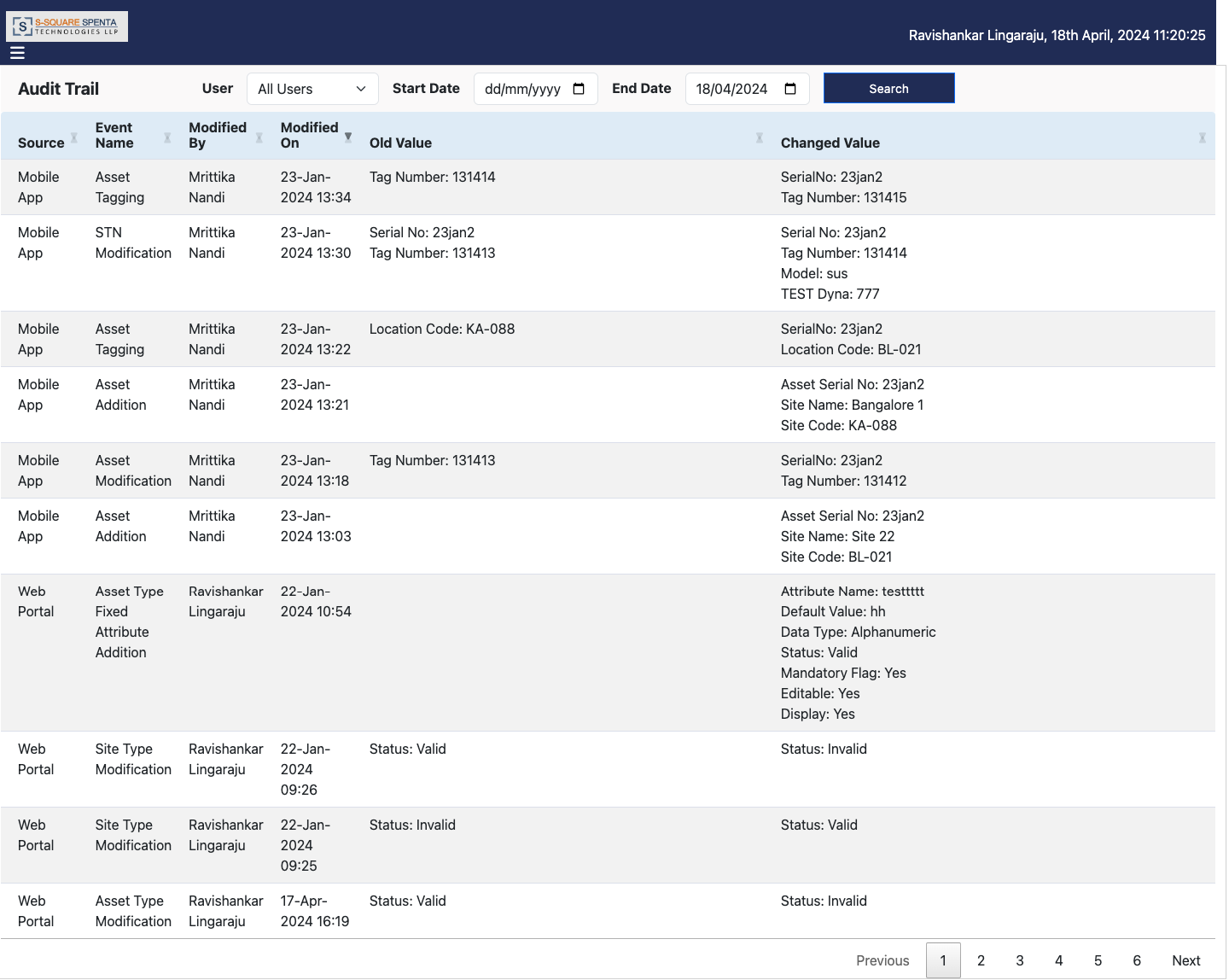
**Output**

* list assets from **t\_asset\_history** table**.**

# LLD: Audit Trail

## View Audit Trail

**Screen Design**

****

**Input** User id using Auth::id();

**Processing**

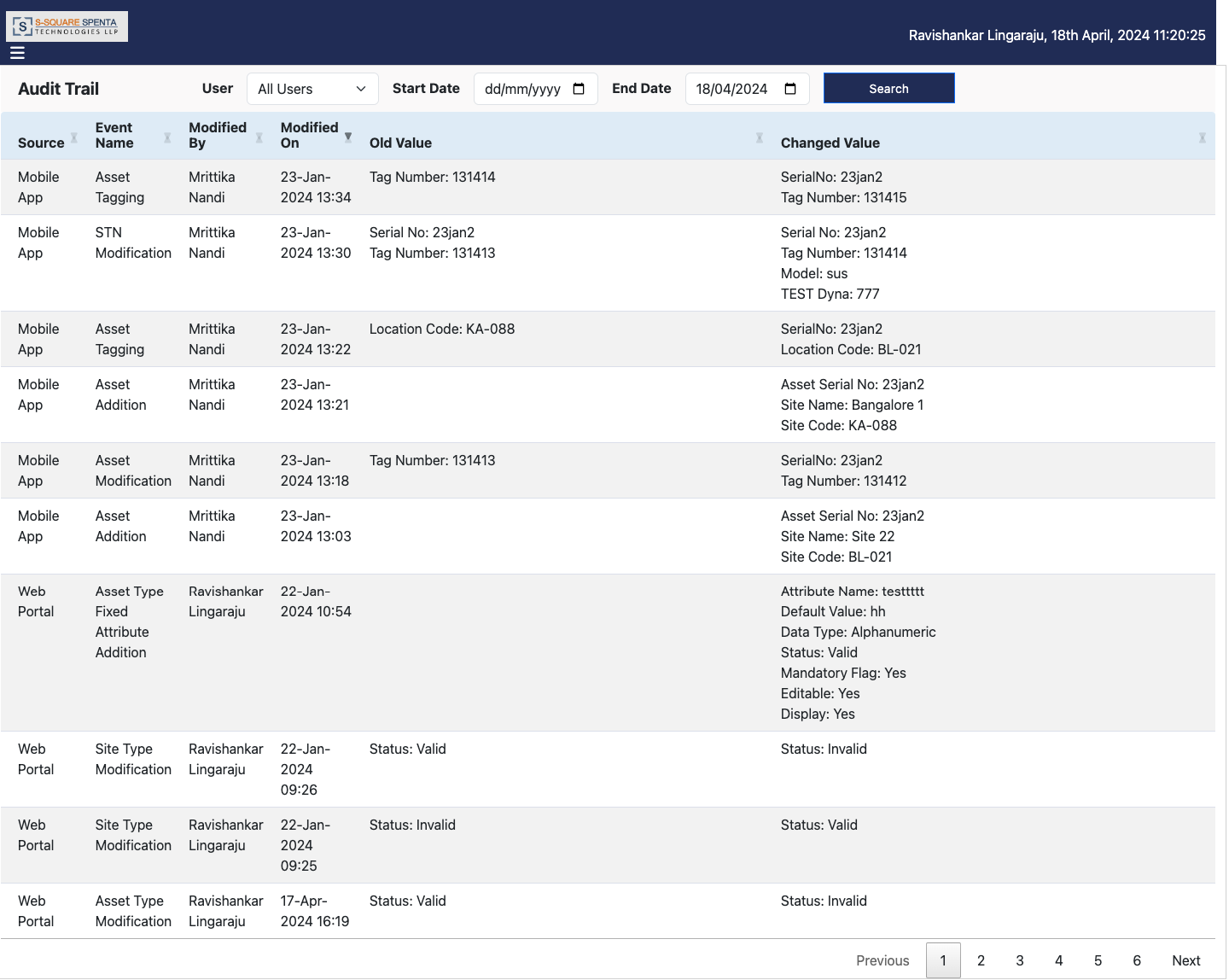
* User clicks Audit Trail from the left menu link.
* Listing of all changes to different tables from **system\_logs** table is listed in descending order by created\_at column with pagination and sourceof changes like mobile app or web portal.
* It also displays the user who changed.

**Output**

* list assets from system\_logs table
* order by created\_at descending

## Search

**Screen Design**

****

**Input**

* **user\_id**: User id of the user who made the changes.
* **start\_date**: Date from which the date to be fetched.
* **end\_date**: The date upto which the data to be fetched

**Processing**

* User clicks Audit Trail from the left menu link.
* The search box shows. Changing the user with start date and end date then clicking search button filters the data from **system\_logs** table.

**Output**

* List assets from system\_logs table based on search options.
* order by **created\_at** descending

# LLD: Process Maker integration

## PROCESSMAKER Login

**Input**

* API endpoint: domain/IP for Processmaker/workflow/oauth2/token

{

"grant\_type": "password",

"scope": "\*",

"client\_id":"x-pm-mobile-client",

"client\_secret":"4426746995afa07e7485df1055471800",

"username":"ravishankar.lingaraju@ssquarespenta.com",

"password": "Password@123"

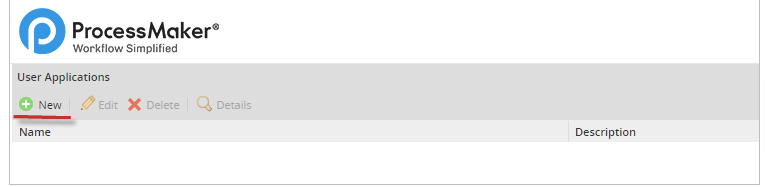
}

**Validations**

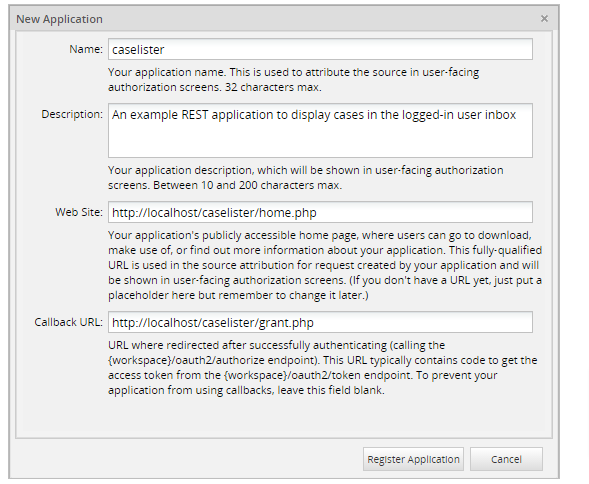
* All the input fields are mandatory.

**Processing**

* For getting the client id and client secret we need the following steps
  + After login to processmaker navigate to [http://{pm-server}/sys{workspace}/en/neoclassic/oauth2/applications](about:blank)

[](about:blank)

* + Click **New** and fill out the following form to register a new application:



* + Once the application has been successfully registered its credentials are generated.

**Output**

{

"access\_token": "29e2e96623c3ee3a0ea1472ce0093e6e22039ce1",

"expires\_in": 86400,

"token\_type": "bearer",

"scope": "\*",

"refresh\_token": "403e497896c02e9620af2e5c494bd3c3c0308dc7"

}

## Project Create

**Input**

**project\_file**: the path to PMX file

**API call: POST** /api/1.0/{workspace}/project/import

**Processing**

* Login to process maker using API
* Import the specific PMX file (Add\_Assets.pmx,Asset\_Tagging.pmx,Audit\_Asset\_Missing.pmx,Audit\_Tag\_Missing.pmx,Edit\_Asset.pmx) using curl request to PM API (/api/1.0/{workspace}/project/import)

**Output**

* Project is created and prj\_uid is returned

{

"prj\_uid": "8131652775363e959e5fdf4087993497"

"prj\_file": "Add\_Assets.pmx"

}

## Pending Approval

**Input**

**Bearer token:**: Access token received from login.

**API call: POST** /api/1.0/workflow/cases/draft

**Processing**

* Login to processmaker using API and get the access token
* Once accesstoken is received use it as bearer token to curl request to /api/1.0/workflow/cases/draft

**Output**

* List of draft cases displayed

[

{

"app\_uid": "50576446953235bfb797531078087088",

"del\_index": "1",

"del\_last\_index": "1",

"app\_number": "1",

"app\_status": "ID\_DRAFT",

"usr\_uid": "00000000000000000000000000000001"

"previous\_usr\_uid": "",

"tas\_uid": "7983935495320c1a75e1df6068322280",

"pro\_uid": "2317283235320c1a36972b2028131767",

"del\_delegate\_date": "2014-03-14 15:43:55",

"del\_init\_date": "2014-03-14 15:43:55",

"del\_task\_due\_date": "2014-03-17 15:43:55",

"del\_finish\_date": null,

"del\_thread\_status": "OPEN",

"app\_thread\_status": "OPEN",

"app\_title": "#1",

"app\_pro\_title": "Invoice Approval Process",

"app\_tas\_title": "Task 1",

"app\_current\_user": " Administrator",

"app\_del\_previous\_user": "",

"del\_priority": "ID\_PRIORITY\_N",

"del\_duration": "0",

"del\_queue\_duration": "0",

"del\_delay\_duration": "0",

"del\_started": "0",

"del\_finished": "0",

"del\_delayed": "0",

"app\_create\_date": "2014-03-14 15:43:55",

"app\_finish\_date": null,

"app\_update\_date": "2014-03-14 15:43:57",

"app\_overdue\_percentage": "0",

"usr\_firstname": "Administrator",

"usr\_lastname": " ",

"usr\_username": "admin",

"appcvcr\_app\_tas\_title": "Task 1",

"usrcr\_usr\_uid": "00000000000000000000000000000001",

"usrcr\_usr\_firstname": "Administrator",

"usrcr\_usr\_lastname": " ",

"usrcr\_usr\_username": "admin"

}

]

## assign task to PM users

**Input**

**Bearer token:**: Access token received from login.

**API call: POST** /api/1.0/workflow/project/{project\_id}/activity/{task\_id}/assignee

**POST Param**

{

'aas\_uid' =>pm\_user\_id,

'aas\_type' => PM\_GROUP\_TYPE

}

**Processing**

* Login to processmaker using API and get the access token
* Once accesstoken is received use it as bearer token to curl request to /api/1.0/workflow/project/{project\_id}/activity/{task\_id}/assignee

**Output**

* Task is assigned to a PM user.
* REsponse from API 201 (Created)