

Asset Management Portal - Final Project Report

1. INTRODUCTION

1.1 Project Overview

The **Asset Management Portal** is a ServiceNow-based solution for efficiently managing physical and digital assets. It enables administrators to track, assign, and maintain assets using a single Asset Inventory table. Key features include automated processes, user-friendly forms, auto-generated asset IDs, and reporting tools. The system improves accuracy, reduces manual work, and supports data-driven decisions through components like UI Actions, and Scheduled Jobs.

1.2 Purpose

The primary purpose of this project is to develop a **centralized and automated platform** for managing organizational assets. It aims to replace manual tracking systems, minimize asset mismanagement, improve lifecycle visibility, and enable proactive actions through automation and reporting in ServiceNow.

2. IDEATION PHASE

2.1 Problem Statement

Many organizations rely on manual methods or disconnected systems to manage their assets, leading to **data inconsistencies, lack of visibility, delayed updates, and inefficient tracking** of asset usage and condition. This results in **asset mismanagement**, increased operational costs, and difficulty in making informed decisions.

2.2 Empathy Map Canvas

Says:

- "I need a centralized place to view and manage all our assets."
- "It's hard to keep track of asset status and assignments manually."

Thinks:

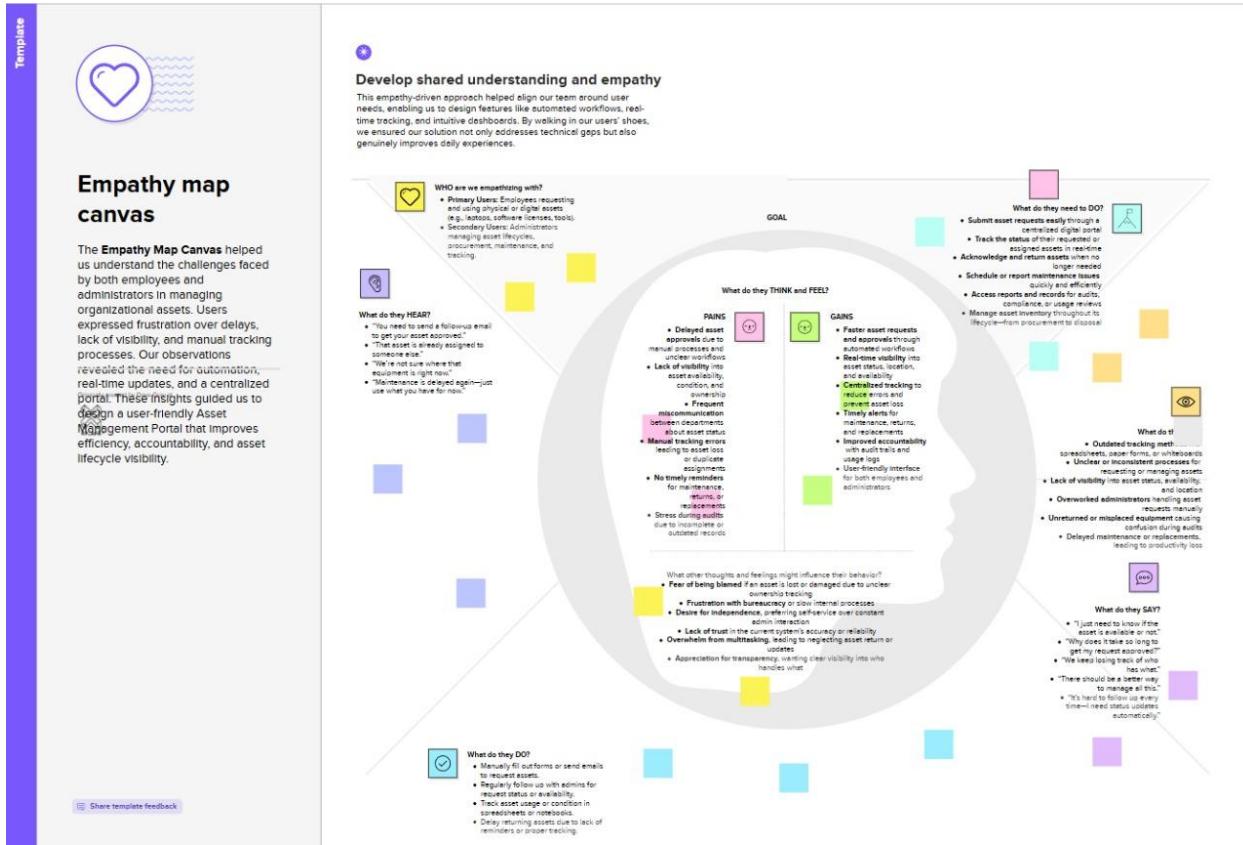
- "Are all our assets being used efficiently?"
- "What if an asset goes missing or isn't maintained on time?"

Does:

- Manually updates spreadsheets or basic records.
- Tracks asset usage by checking with teams or physical inventories.

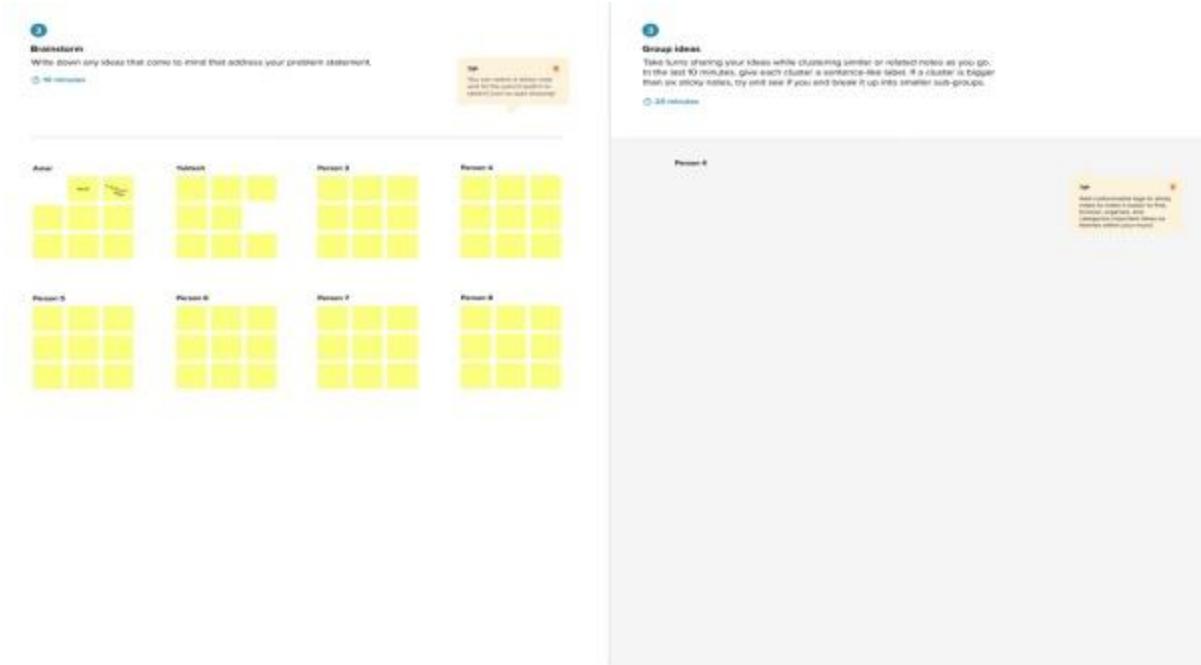
Feels:

- Overwhelmed by the complexity of managing numerous assets.
- Frustrated with errors, delays, and lack of visibility in the current system.



2.3 Brainstorming

The team explored ideas such as using a custom ServiceNow table (Asset Inventory) to manage all asset records, configuring auto-numbering with a prefix for easy identification, and implementing UI Actions to streamline asset status updates (e.g., “Mark as Repaired”). Discussions also included setting up scheduled jobs for maintenance alerts, and integrating reports and dashboards for better visibility. The focus was on building a centralized, user-friendly, and scalable system to automate and simplify asset management.



3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

Users create and update asset records, assign them to employees, and track their lifecycle stages such as repair or disposal. The system provides automation for status changes, maintenance alerts, and visual insights through reports and dashboards.

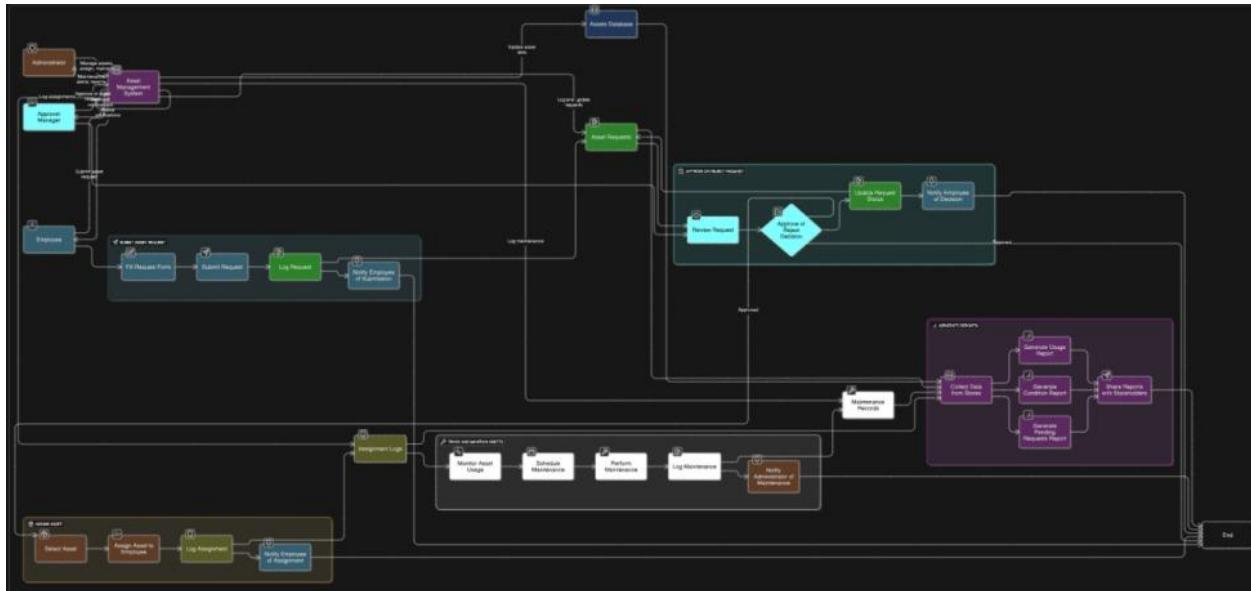
3.2 Solution Requirement

- Single **Asset Inventory** table to manage all asset data
- **UI Actions** for quick lifecycle operations like "Mark as Repaired"

- **Scheduled Jobs** for warranty expiry or maintenance notifications
- **Reporting and Dashboard capability** for asset tracking and performance insights

3.3 Data Flow Diagram

The DFD shows data flowing from user forms → validation → storage in tables → automation triggers → reports/alerts generation.



3.4 Technology Stack

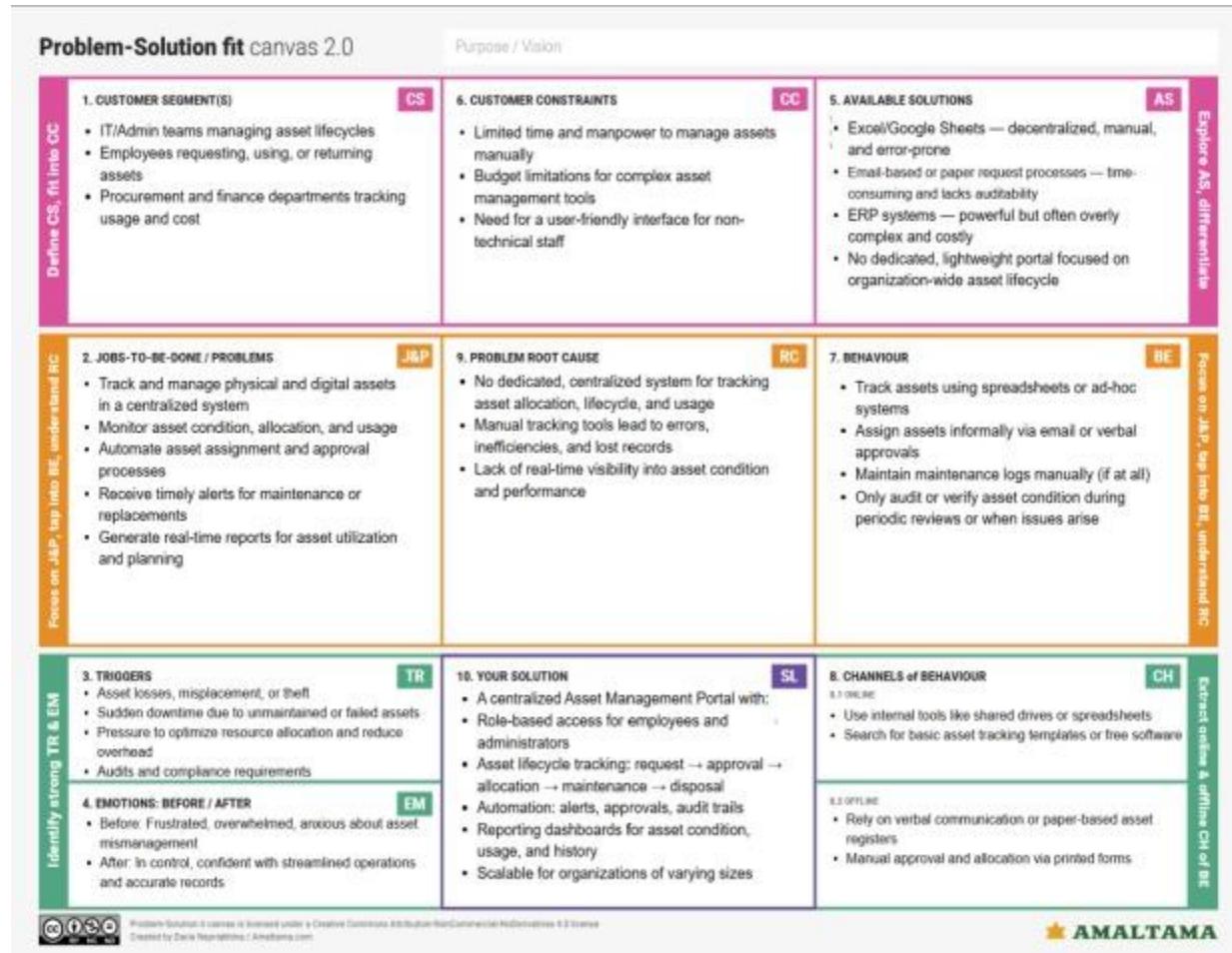
- ServiceNow Custom Tables and Forms
- Glide API, Business Rules, UI Policies
- ServiceNow Notification Engine
- MySQL Backend (ServiceNow-managed)
- ServiceNow REST APIs (optional for future integrations)

4. PROJECT DESIGN

4.1 Problem–Solution Fit

The system addresses the challenges of manual and fragmented asset tracking by providing a **centralized platform** for managing all organizational assets. It simplifies the process of recording, assigning, and maintaining assets while automating routine tasks such as status

updates and maintenance alerts. With features like auto-generated asset IDs, UI actions, and reporting, the portal ensures improved visibility, accuracy, and control over the entire asset lifecycle.



4.2 Proposed Solution

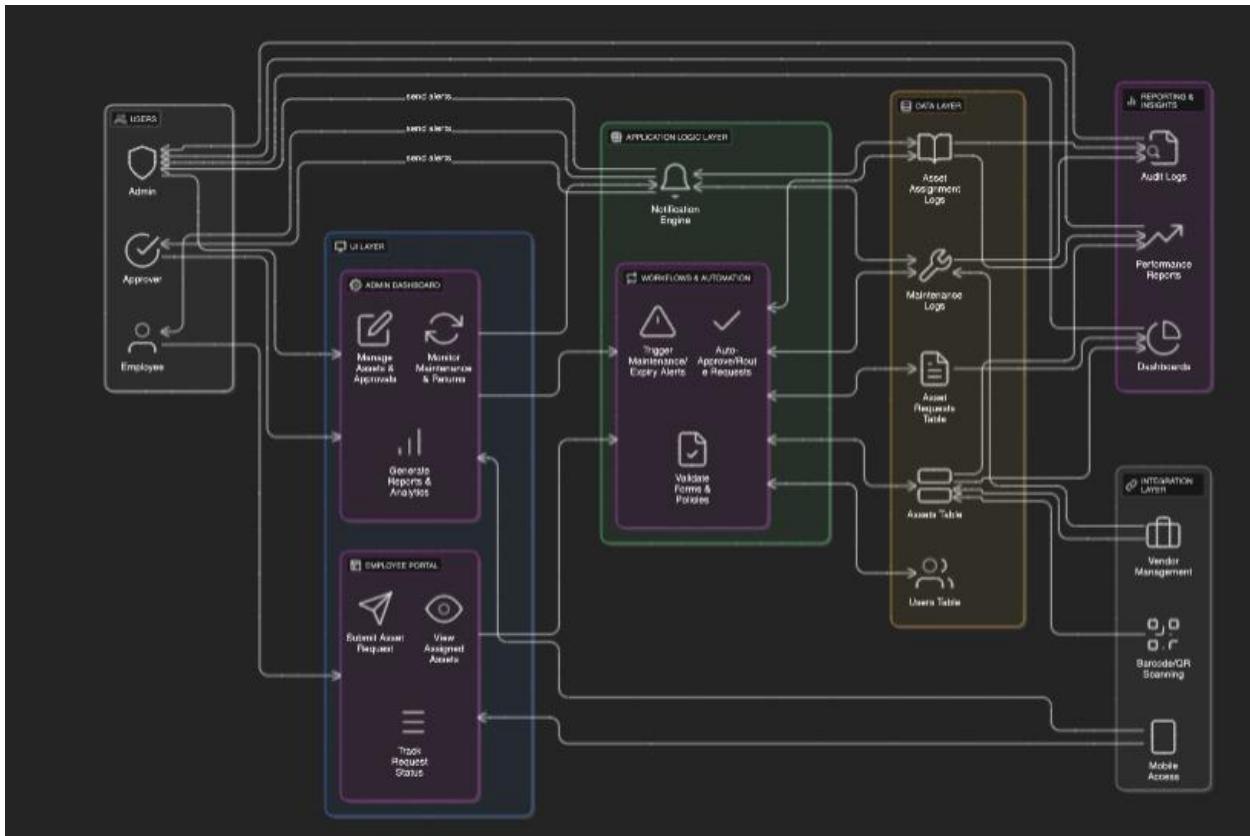
A ServiceNow-based tool with:

- Custom tables
- UI Actions
- Scheduled Jobs
- Categorized reports
- Testing UI Actions and Scheduled Job

4.3 Solution Architecture

The architecture comprises four layers:

- **Data Layer:** Stores all asset details in a single Asset Inventory table.
- **Logic Layer:** Handles automation via Business Rules, UI Actions, Scheduled Jobs, and Number Maintenance.
- **UI Layer:** Provides user-friendly forms, UI Policies, and related lists for effective interaction.
- **Configuration Layer:** Uses Update Sets for deploying system components and configurations



5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

The project was completed over 3 sprints:

- **Sprint 1:** Instance setup, update set creation, Asset Inventory table setup (8 points)
- **Sprint 2:** Form customization, auto-numbering, business rules, and UI actions (7 points)
- **Sprint 3:** Scheduled jobs, maintenance alert setup, and report/dashboard creation (5 points)

Velocity: 20 story points / 3 sprints = ~6.67 points per sprint

The Project was completed as the following milestones covering 3 sprints

The team executed these milestones:

1. ServiceNow Instance Setup
 - Signed up at developer.servicenow.com and requested a Personal Developer Instance (PDI)
 - Filled necessary details; received instance access credentials via email
 - Logged in and prepared the instance for development
2. Creation of Asset Inventory Table
 - Created the Asset Inventory table .
 - Create the following fields:
 - i. Assigned to : string
 - ii. Status : choice
 - iii. Purchase date : date
 - iv. Warranty Expire : date
 - v. Asset name : string
 - vi. Type : choice

Number : String

Asset Inventory Table Fields

Field Name	Type	
Number	String	Auto populate Number with Prefix ASSET
Status	Choice	
Assigned to	String	
Status	Choice	
Purchase Expire	Date	

Column label	Type	Reference	Max length	Default value	Display
Type	Choice	(empty)	40	false	
Status	Choice	(empty)	40	false	
Created	Date/Time	(empty)	40	false	
Assigned to	String	User	40	false	
Asset name	String	(empty)	40	false	
Sys ID	Sys ID (GUID)	(empty)	32	false	
Updated by	String	(empty)	40	false	
Updates	Integer	(empty)	40	false	
Updated	Date/Time	(empty)	40	false	
Number	String	(empty)	40	false	
Purchase date	Date	(empty)	40	false	
Created by	String	(empty)	40	false	
Warranty Expire	Date	(empty)	40	false	

3. Creation of UI Actions

- The "**Mark As Lost**" UI Action in the Asset Inventory table enables users to update an asset's status to "Lost" with one click.
- It includes a condition to ensure appropriate visibility and uses a script to automate the update and redirect. .

The screenshot shows the 'UI Action - New Record' configuration page for a new record. The action is named 'Mark As Lost' and is associated with the 'Asset Inventory [u_asset_inventory]' table. The order is set to 100, and the action name is 'mark_as_lost'. The 'Active' checkbox is checked, and 'Show insert' and 'Show update' are also checked. The 'Client' checkbox is unchecked. The 'Overrides' field is empty. The 'Application' dropdown is set to 'Global'. The 'Form button' checkbox is checked. The 'Form context menu' checkbox is unchecked. The 'Form link' checkbox is unchecked. The 'List banner button' checkbox is unchecked. The 'List bottom button' checkbox is unchecked. The 'List context menu' checkbox is unchecked. The 'List choice' checkbox is unchecked. The 'List link' checkbox is unchecked. The 'List style' dropdown is set to 'None'. The 'Messages' and 'Comments' fields are empty. The 'Hint' field is empty. The 'Condition' field contains the expression 'current.u_status != "Lost"'. The 'Script' section contains the following code:

```

Turn on ECMAScript 2021 (ES12) mode ⓘ
1 current.u_status = "Lost";
2 current.update();
3 action.setRedirectURL(current);

```

The 'Protection policy' dropdown is set to 'None'. At the bottom, there are 'Workspace' and 'Requires role' buttons.

4. Creation of Second UI Action

- The "**Mark As Repaired**" UI Action was implemented in the Asset Inventory table to simplify restoring asset status.
- It becomes available when an asset is marked as **Damaged** or **Lost**, allowing users to update its status to **Available** with a single click.

The screenshot shows the 'UI Action - New Record' configuration page for a new record. The action is named 'Mark As Repaired' and is associated with the 'Asset Inventory [u_asset_inventory]' table. The order is set to 100, and the action name is 'mark_as_repaired'. The 'Active' checkbox is checked, and 'Show insert' and 'Show update' are also checked. The 'Client' checkbox is unchecked. The 'Overrides' field is empty. The 'Application' dropdown is set to 'Global'. The 'Form button' checkbox is unchecked. The 'Form context menu' checkbox is unchecked. The 'Form link' checkbox is unchecked. The 'List banner button' checkbox is unchecked. The 'List bottom button' checkbox is unchecked. The 'List context menu' checkbox is unchecked. The 'List choice' checkbox is unchecked. The 'List link' checkbox is unchecked. The 'List style' dropdown is set to 'None'. The 'Messages' and 'Comments' fields are empty. The 'Hint' field is empty. The 'Condition' field contains the expression 'current.u_status == "Damaged" || current.u_status == "Lost"'. The 'Script' section contains the following code:

```

Turn on ECMAScript 2021 (ES12) mode ⓘ
1 current.u_status = "Available";
2 current.update();
3 action.setRedirectURL(current);

```

The 'Protection policy' dropdown is set to 'None'. At the bottom, there are 'Workspace' and 'Requires role' buttons.

5. Third UI Action Creation

- The "**Mark As Damaged**" UI Action was created in the Asset Inventory table to quickly update an asset's condition when it is found to be damaged.

- This action is only available if the asset is not already marked as **Damaged**.

)Creation of Scheduled Jobs

- The "**Warranty Expiry Alert**" **Scheduled Job** was created to automatically check for assets nearing the end of their warranty period. Configured to run **daily at 12:00 PM**, the job uses a script to identify such assets and can be extended to send notifications or trigger maintenance actions.
- Implemented the logic to manage data consistency or automate actions (actual script written as part of development)

```

var grAsset = new GlideRecord('u_asset_inventory'); // Replace with your table name
var today = new GlideDateTime();
var futureDate = new GlideDateTime();
futureDate.addDays(30); // Get date 30 days from now
grAsset.addQuery('u_warranty_expire', '<=', futureDate); // Warranty expiring within the next 30 days
grAsset.addQuery('u_warranty_expire', '>=', today); // Warranty expiring after today
grAsset.query();
while (grAsset.next()) {
    var email = new GlideEmailOutbound();
    email.setSubject("Warranty Expiry Alert: " + grAsset.getValue('u_assest_name')); // Use getValue for dynamic field access
    email.setBody("The warranty for " + grAsset.getValue('u_assest_name') + " (Type: " + grAsset.getValue('u_asset_type') +
        ") is expiring soon on " + grAsset.getValue('u_warranty_expiry') + ". Please take action."); // Get values dynamically
    email.setTo('it-support@company.com'); // Change to your IT support email
    email.send();
}

```

```

        gs.info("Email sent for asset: " + grAsset.getValue('u_assest_name')); // Log for confirmation
    }
}

```

The screenshot shows the 'Scheduled Script Execution - New Record' page in ServiceNow. The script is titled 'Warranty Expiry Alert'. It is set to run daily at 12:00. The script code uses GlideScript to query assets with expiring warranties and send an email to the asset owner.

```

Name: Warranty Expiry Alert
Active: checked
Application: Global
Conditional: none

Run: Daily
Time zone: None
Time (Hours): 12:00

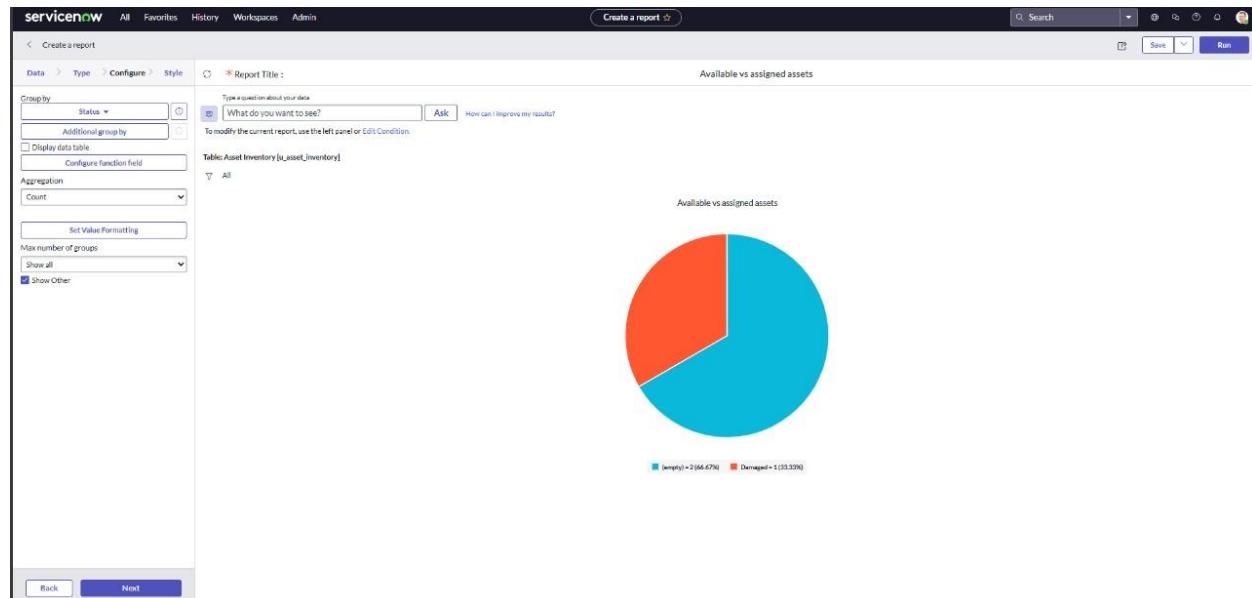
Run this script: (checkbox) Turn on ECMAScript(2021 ES12) mode (radio)

1 var grasset = new GlideRecord('u_asset_inventory'); // Replace with your table name
2
3 var today = new GlideDateTime();
4
5 var futureDate = new GlideDateTime();
6
7 futureDate.addDays(30); // set date 30 days from now
8
9 grasset.addQuery('u_warranty_expire', '<=', futureDate); // warranty expiring within the next 30 days
10 grasset.addQuery('u_warranty_expire', '>=', today); // warranty expiring after today
11
12 grasset.query();
13
14 while (grasset.next()) {
15
16     var email = new GlideEmail();
17
18     email.setSubject('Warranty Expiry Alert: ' + grasset.getValue('u_assest_name')) // Use getValue for dynamic field access
19
20     email.setBody("The warranty for " + grasset.getValue('u_assest_name') + " (" + grasset.getValue('u_assest_type') + "


```

6. Creation of Reports

The **"Available vs Assigned Assets"** report was created in ServiceNow to provide a visual summary of asset distribution based on their status. Using the **Asset Inventory** table as the data source, the report is configured as a **Pie Chart**, grouped by the **Status** field with a **Count** aggregation.



6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

The system was tested for:

- Accurate **update of asset status** using UI Actions
- Timely execution of **Scheduled Jobs** for warranty and maintenance alerts
- Proper **form behavior** based on UI Policies (e.g., conditionally visible fields)
- Reliable display of **report data** grouped by asset status
- Efficient loading and updating of **Asset Inventory records**

7. RESULTS

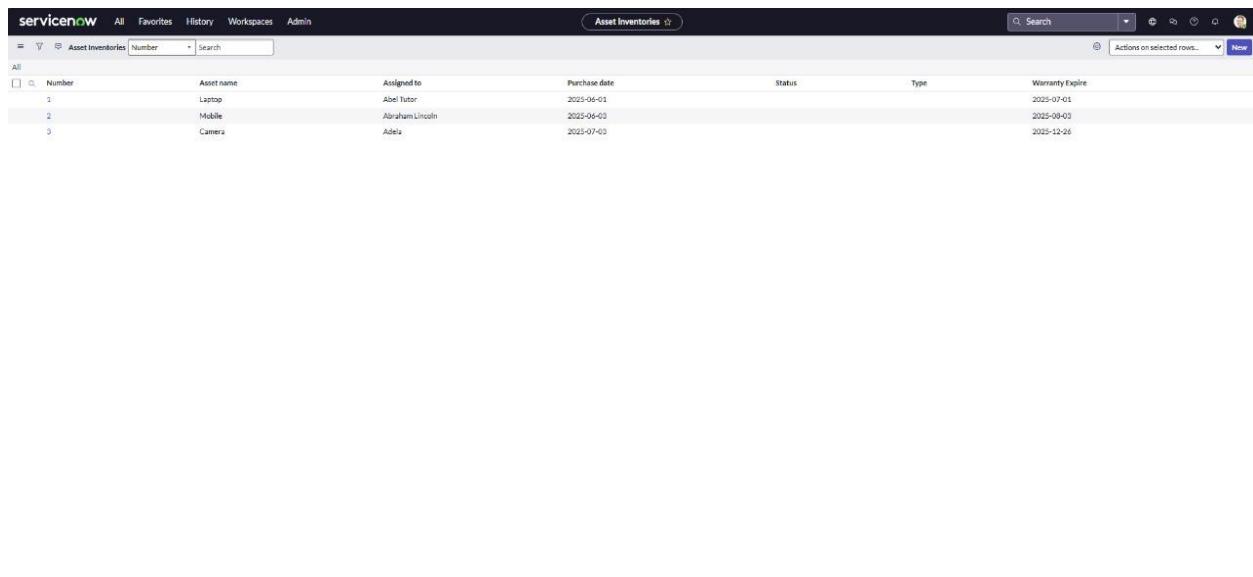
7.1 Output Screenshots

- Tested UI action.

A screenshot of the ServiceNow Asset Inventory form. The top navigation bar includes 'servicenow' and links for 'All', 'Favorites', 'History', 'Workspaces', and 'Admin'. The title bar says 'Asset Inventory'. The main form has the following fields:

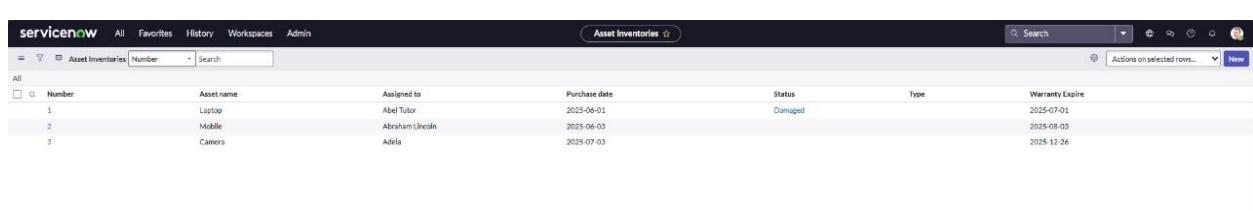
- Number: 1
- Assigned to: Abel Tutor
- Status: None
- Purchase date: 2025-06-01
- Warranty Expire: 2025-07-01
- Asset name: Laptop
- Type: None

At the bottom of the form are buttons for 'Update', 'Mark As Lost', 'Mark As Repaired', and 'Delete'.



The screenshot shows a ServiceNow Asset Inventories grid with the following data:

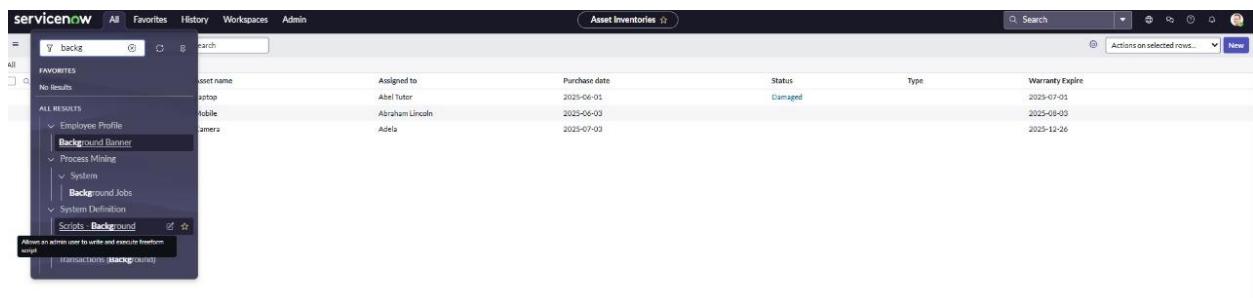
Number	Asset name	Assigned to	Purchase date	Status	Type	Warranty Expire
1	Laptop	Abel Tutor	2025-06-01	Active	Computer	2025-07-01
2	Mobile	Abraham Lincoln	2025-06-03	Active	Smartphone	2025-09-03
3	Camera	Adela	2025-07-03	Active	Photograph	2025-12-26



The screenshot shows a ServiceNow Asset Inventories grid with the following data. The second row is highlighted.

Number	Asset name	Assigned to	Purchase date	Status	Type	Warranty Expire
1	Laptop	Abel Tutor	2025-06-01	Active	Computer	2025-07-01
2	Mobile	Abraham Lincoln	2025-06-03	Active	Smartphone	2025-09-03
3	Camera	Adela	2025-07-03	Active	Photograph	2025-12-26

- Tested Scheduling Job.



The screenshot shows a ServiceNow search results page for the term "background". The left sidebar shows a navigation tree with the following structure:

- FAVORITES
 - No Results
- ALL RESULTS
 - Employee Profile
 - Process Mining
 - System
 - System Definition
 - Scripts
 - Background

The search results table shows the following data:

Number	Asset name	Assigned to	Purchase date	Status	Type	Warranty Expire
1	Laptop	Abel Tutor	2025-06-01	Active	Computer	2025-07-01
2	Mobile	Abraham Lincoln	2025-06-03	Active	Smartphone	2025-09-03
3	Camera	Adela	2025-07-03	Active	Photograph	2025-12-26

A tooltip at the bottom left of the sidebar says: "Allow an admin user to write and execute freetext script transactions [stacked query]."

The screenshot shows the ServiceNow script editor interface. At the top, there's a navigation bar with 'servicenow' and other tabs like 'All', 'Favorites', 'History', 'Workspaces', and 'Admin'. Below the bar, a message says 'Running freeform script can cause system disruption or loss of data.' A warning 'Run script [Java script] executed on server.' is also present. The main area contains a Java script code block:

```

1 var grasset = new GlideRecord('u_asset_inventory'); // Replace with your table name
2 var today = new GlideDateTime();
3 var futuredate = new GlideDateTime();
4 futuredate.addDays(30); // Get date 30 days from now
5
6 grasset.query("u_warranty_expire", "<", futuredate); // Warranty expiring within the next 30 days
7 grasset.addQuery("u_warranty_expire", ">", today); // Warranty expiring after today
8 grasset.query();
9
10 while (grasset.next()) {
11     var email = new GlideEmail();
12     email.setSubject("Warranty Expire Alert: " + grasset.getValue('u_asset_name')); // Use getvalue for dynamic field access
13     email.setBody("The warranty for " + grasset.getValue('u_asset_name') + " (" + grasset.getValue('u_asset_type') +
14         ") is expiring soon on " + grasset.getValue('u_warranty_expire') + ". Please take action."); // Set values dynamically
15
16     email.setTo('it-support@company.com'); // Change to your IT support email
17     email.send();
18
19     gs.info("Email sent for asset: " + grasset.getValue('u_asset_name')) // Log for confirmation
20 }
21

```

Below the code, there are several execution options: 'Run Script', 'In scope [global]', 'Record for rollback', 'Execute in sandbox', 'Execute as scriptlet', and 'Cancel after 4 hours'. A note '+ Instance Scripts' is also visible.

The screenshot shows the ServiceNow log viewer. At the top, it displays 'servicenow' and other tabs. Below the tabs, a message says '[00:00:368] Script completed in scope global: asset'. It also shows 'Script execution history and recovery' with a link 'available here'. The main area contains the log output:

```

*** script: Email sent for asset: null

```

8. ADVANTAGES & DISADVANTAGES

Advantages

- Centralized, automated expense tracking
- Real-time budget monitoring
- Easy to extend for more features
- Low-code development for rapid deployment

Disadvantages

- Requires ServiceNow knowledge for configuration
- Depends on PDI availability or enterprise licensing

9. CONCLUSION

The **Asset Management Portal** offers an end-to-end solution for efficient asset tracking and management using ServiceNow. It leverages automation, real-time updates, and reporting to reduce asset downtime, improve accountability, and support data-driven decisions. By streamlining workflows and enhancing visibility, the system helps organizations maximize asset value, reduce costs, and boost productivity.

10. FUTURE SCOPE

- Integration with external financial planning tools
- Advanced analytics and dashboards
- Mobile-friendly forms for easier data entry
- Multi-family or community-level expense tracking

11. APPENDIX

Video Demo link:

<https://drive.google.com/file/d/19PESXjqhSJp6JLZLAKOvRrNxfiMTf5Bz/view?usp=sharing>