

Asset Management Portal

Team ID: LTVIP2026TMIDS47869

Team Size:5

Team Leader: Siripireddy Indira Reddy

Team member: Kota Chandana

Team member: Suram Pallavi

Team member: Lachaiahgari Vaishnavi

Team member: N.P. Ganga Devi

Asset Management Portal

Problem Statement:

The Asset Management Portal will streamline the tracking, management, and allocation of both physical and digital assets across an organization. Employees will be able to request and receive assets through an intuitive portal, while administrators can manage the entire asset lifecycle, from procurement to disposal. The portal will also automate asset assignment, ensure accurate record-keeping, and generate real-time reports on asset utilization and condition. Alerts will be triggered for maintenance or replacement needs, ensuring optimal asset performance and reducing downtime. By centralizing asset management, the platform will improve operational efficiency, reduce asset loss, and support informed decision-making.

INDEX:

1. Introduction

- 1.1 Project Overview
- 1.2 Purpose of the Project

2. Problem Statement

3. System Analysis

- 3.1 Existing System
- 3.2 Proposed System
- 3.3 System Requirements
 - 3.3.1 Functional Requirements
 - 3.3.2 Non-Functional Requirements

4. System Design

- 4.1 Architecture Diagram
- 4.2 Database Design (Asset Inventory Table)
- 4.3 Field Description

5. Implementation

- 5.1 Table Creation
- 5.2 Field Creation
- 5.3 UI Actions
 - 5.3.1 Mark As Lost
 - 5.3.2 Mark As Damaged
 - 5.3.3 Mark As Repaired
- 5.4 Scheduled Job – Warranty Expiry Alert
- 5.5 Report Creation – Available vs Assigned Assets

6. Testing

- 6.1 UI Action Testing
- 6.2 Scheduled Job Testing

7. Results

8. Advantages

9. Conclusion

10.Future Enhancements

11. **Appendix**

11.1 Source Code

11.2 Screenshots

11.3 References

1. INTRODUCTION

1.1 Project Overview

The Asset Management Portal is a web-based application developed using the ServiceNow platform to efficiently manage and track organizational assets throughout their lifecycle. The system provides a centralized solution for maintaining records of both physical and digital assets, including their allocation, status updates, warranty details, and maintenance tracking.

In this project, a custom table named *Asset Inventory* was created to store asset information. UI Actions were implemented to update asset status dynamically (such as marking assets as Lost, Damaged, or Repaired). A Scheduled Job was developed to automatically generate warranty expiry alerts. Additionally, reports were created to visualize asset availability and assignment distribution.

The system automates asset tracking processes, improves visibility, and reduces manual errors by leveraging ServiceNow's built-in automation and reporting features.

1.2 Purpose of the Project

The main purpose of the Asset Management Portal is to streamline asset tracking and lifecycle management within an organization.

The objectives of this project are:

- ☐ To centralize asset information in a structured database.
- ☐ To automate asset status updates using UI Actions.
- ☐ To generate automatic warranty expiry alerts using Scheduled Jobs.
- ☐ To provide real-time asset reports for better decision-making.
- ☐ To improve operational efficiency and reduce asset mismanagement.

By implementing this system on the ServiceNow platform, organizations can enhance asset accountability, minimize downtime, and ensure proactive maintenance management.

2. PROBLEM STATEMENT

In many organizations, asset management is handled manually using spreadsheets, emails, or paper-based records. This traditional approach leads to several challenges such as inaccurate data, lack of real-time visibility, delayed maintenance tracking, and difficulty in monitoring asset lifecycle status.

Without a centralized system, administrators face problems in tracking which assets are available, assigned, damaged, or lost. Employees may experience delays in asset allocation due to inefficient approval processes. Additionally, there is no automated mechanism to track warranty expiration, which may result in missed maintenance opportunities and increased operational costs.

The absence of structured reporting also makes it difficult for management to analyze asset utilization and make informed decisions.

To overcome these challenges, there is a need for an automated and centralized Asset Management Portal that:

- ☐ Tracks assets throughout their lifecycle.
- ☐ Updates asset status dynamically.
- ☐ Generates warranty expiry alerts automatically.
- ☐ Provides real-time reporting and analytics.
- ☐ Improves transparency and accountability.

The proposed solution leverages the ServiceNow platform to automate asset tracking, enhance operational visibility, and optimize asset utilization within the organization.

3. SYSTEM ANALYSIS

System analysis involves studying the existing system, identifying its limitations, and defining the requirements for the proposed system.

3.1 Existing System

In many organizations, asset management is handled manually through spreadsheets, emails, or basic record-keeping systems. The existing system has several limitations:

- ☐ Manual data entry increases chances of errors.
- ☐ No real-time tracking of asset status.
- ☐ Difficulty in identifying assets nearing warranty expiry.
- ☐ No automated alerts or notifications.
- ☐ Lack of centralized reporting.
- ☐ Delays in updating asset status (Lost, Damaged, Repaired).

Because of these limitations, asset monitoring becomes inefficient, leading to asset misplacement, increased downtime, and poor decision-making.

3.2 Proposed System

The proposed system is a ServiceNow-based Asset Management Portal that automates asset lifecycle management.

Key features of the proposed system:

- ☐ Centralized Asset Inventory table.
- ☐ UI Actions to update asset status dynamically.
- ☐ Automated Scheduled Job for warranty expiry alerts.
- ☐ Real-time reports (Available vs Assigned assets).
- ☐ Role-based access and controlled updates.
- ☐ Improved data accuracy and operational transparency.

This system eliminates manual tracking and improves efficiency by leveraging ServiceNow automation capabilities.

3.3 System Requirements

3.3.1 Functional Requirements

The system must:

- ☐ Allow administrators to create and manage asset records.
- ☐ Store asset details such as name, type, assigned user, status, and warranty expiry.
- ☐ Provide UI buttons to:
 - Mark asset as Lost
 - Mark asset as Damaged
 - Mark asset as Repaired
- ☐ Automatically check for warranty expiry using a Scheduled Job.
- ☐ Send email alerts for assets expiring within 30 days.
- ☐ Generate reports to display asset status distribution.
- ☐ Allow testing via Background Scripts.

3.3.2 Non-Functional Requirements

The system should:

- ☐ Ensure data accuracy and consistency.
- ☐ Provide secure access control.
- ☐ Perform scheduled jobs efficiently without delay.
- ☐ Generate reports quickly.
- ☐ Maintain system reliability and availability.

4. SYSTEM DESIGN

System design defines the structure, architecture, and data organization of the Asset Management Portal implemented in ServiceNow.

4.1 Architecture Diagram

The system follows a simple ServiceNow-based architecture:

User (Administrator)



ServiceNow Form (Asset Inventory Table)



UI Actions / Scheduled Job



ServiceNow Database



Reports & Notifications

Explanation:

- ☐ The administrator interacts with the Asset Inventory form.
- ☐ UI Actions update asset status dynamically.
- ☐ The Scheduled Job runs daily to check warranty expiry.
- ☐ Data is stored in the custom table.
- ☐ Reports generate visual insights from stored data.

This architecture ensures centralized control and automation within the ServiceNow platform.

4.2 Database Design (Asset Inventory Table)

Table Name:

Asset Inventory

Table Label:

Asset Inventory

Purpose:

To store complete details of all organizational assets.

Primary Fields in Table:

| Field Name | Field Type | Description |
|-------------------|------------|---|
| u_asset_name | String | Name of the asset |
| u_asset_type | Choice | Type of asset (Laptop, Desktop, etc.) |
| assigned_to | | |
| u_status | Choice | Asset status (Available, Lost, Damaged) |
| u_purchase_date | Date | Purchase date |
| u_warranty_expire | Date | Warranty expiry date |
| | | |
| | | |
| | | |

This structured table allows efficient asset tracking and reporting.

4.3 Field Description

1. Asset Name

Stores the name of the asset (e.g., Laptop, Printer).

2. Asset Type

A choice field to categorize asset types.

3. Assigned To

Reference field linking to the User table to track asset ownership.

4. Status

Choice field used to track asset condition:

- ☐ Available
- ☐ Lost
- ☐ Damaged

5. Purchase Date

Stores the date when asset was purchased.

6. Warranty Expiry

Used in Scheduled Job to trigger warranty alert emails.

5. IMPLEMENTATION

This section explains how the Asset Management Portal was implemented in the ServiceNow platform, including table creation, UI Actions, Scheduled Job, and report generation.

5.1 Table Creation

A custom table named **Asset Inventory** was created in ServiceNow.

Steps Followed:

1. Navigate to **System Definition** → **Tables**
2. Click on **New**
3. Provide:
 - Table Label: Asset Inventory
 - Table Name: u_asset_inventory
4. Click **Submit**

Purpose:

The table stores all asset-related information such as asset name, type, assigned user, status, purchase date, and warranty expiry date.

5.2 Field Creation

Custom fields were added to the Asset Inventory table to capture necessary asset details.

Fields Created:

- ☐ Asset Name (String)
- ☐ Asset Type (Choice)
- ☐ Assigned To (Reference – User table)
- ☐ Status (Choice: Available, Lost, Damaged)
- ☐ Purchase Date (Date)
- ☐ Warranty Expiry (Date)

Purpose:

These fields allow structured asset data storage and enable automation features like status updates and warranty alerts.

5.3 UI Actions

UI Actions were created to dynamically update asset status directly from the form view.

5.3.1 UI Action – Mark As Lost

- ☐ Table: Asset Inventory
- ☐ Action Name: mark_as_lost
- ☐ Condition:
- ☐ `current.u_status != 'Lost'`

Script:

```
current.u_status = 'Lost';  
  
current.update();  
  
action.setRedirectURL(current);
```

Function:

Changes asset status to "Lost" when clicked.

5.3.2 UI Action – Mark As Damaged

- ☐ Table: Asset Inventory
- ☐ Action Name: mark_as_damaged
- ☐ Condition:
- ☐ `current.u_status != 'Damaged'`

Script:

```
current.u_status = 'Damaged';  
  
current.update();  
  
action.setRedirectURL(current);
```

Function:

Marks the asset as "Damaged" for maintenance tracking.

5.3.3 UI Action – Mark As Repaired

- ☐ Table: Asset Inventory

- ☐ Action Name: mark_as_repaired
- ☐ Condition:
- ☐ `current.u_status == 'Damaged' || current.u_status == 'Lost'`

Script:

```
current.u_status = 'Available';
```

```
current.update();
```

```
action.setRedirectURL(current);
```

Function:

Updates asset status back to "Available" after repair.

5.4 Scheduled Job – Warranty Expiry Alert

A Scheduled Job was created to automatically check for assets whose warranty expires within 30 days.

Configuration:

- ☐ Navigate to: System Definition → Scheduled Jobs
- ☐ Name: Warranty Expiry Alert
- ☐ Run: Daily
- ☐ Time: 12:00 PM

Purpose:

To automatically send email notifications to IT support for assets nearing warranty expiry.

The script uses GlideRecord to fetch assets and GlideEmailOutbound to send email alerts.

5.5 Report Creation

A visual report was created to analyze asset distribution.

Report Details:

- ☐ Report Name: Available vs Assigned Assets
- ☐ Source Type: Table
- ☐ Table: Asset Inventory
- ☐ Type: Pie Chart

- ☐ Group By: Status
- ☐ Aggregation: Count

Purpose:

Provides graphical representation of asset availability and usage.

6. TESTING

Testing ensures that all implemented functionalities work correctly and meet system requirements.

6.1 UI Action Testing

The UI Actions were tested to verify correct status updates.

Testing Steps:

1. Navigate to **Asset Inventory** table.
2. Click on **New**.
3. Enter asset details:
 - Asset Name: Laptop
 - Asset Type: Laptop
 - Assigned To: Abel Tutor
 - Status: Available
 - Select purchase date and warranty expiry date.
4. Click **Submit**.
5. Open the saved record.
6. Click on **Mark As Lost** button.
7. Verify that status changes to **Lost**.
8. Click on **Mark As Repaired**.
9. Verify that status changes back to **Available**.
10. Click on **Mark As Damaged**.
11. Verify that status changes to **Damaged**.

Result:

All UI Actions successfully updated the asset status as expected.

6.2 Scheduled Job Testing

The Scheduled Job was tested to ensure warranty alerts are triggered correctly.

Testing Steps:

1. Create a test asset with warranty expiry date within 30 days.
2. Navigate to **Background Scripts**.
3. Paste the Scheduled Job script.
4. Click **Run Script**.
5. Check:
 - Email sent confirmation.
 - System logs (gs.info output).

Result:

Email notifications were successfully triggered for assets nearing warranty expiry.

6.3 Report Testing

The generated report was tested for accuracy.

Steps:

1. Navigate to **Reports**.
2. Open “Available vs Assigned Assets”.
3. Click **Run**.
4. Verify pie chart values match database records.

Result:

Report correctly displays asset distribution by status.

7. RESULTS

The Asset Management Portal was successfully implemented using the ServiceNow platform.

All core functionalities such as asset creation, status updates, warranty alerts, and reporting were tested and verified.

Key Outcomes:

- The custom **Asset Inventory** table effectively stores and manages asset data.
- UI Actions allow administrators to dynamically update asset status (Lost, Damaged, Repaired).
- The Scheduled Job automatically detects assets with warranty expiry within 30 days.
- Email notifications are triggered successfully for proactive maintenance.
- The Pie Chart report provides a clear visual representation of asset distribution.
- Data updates are reflected in real-time within the system.

The system improves operational efficiency by automating manual processes and reducing errors. It ensures better asset tracking, accountability, and decision-making support for administrators.

8. ADVANTAGES

The Asset Management Portal provides several advantages to the organization by automating and centralizing asset tracking processes.

1. Centralized Asset Management

All asset information is stored in a single structured table, improving visibility and organization.

2. Automated Status Updates

UI Actions allow quick and accurate updates of asset conditions such as Lost, Damaged, or Repaired.

3. Proactive Warranty Monitoring

The Scheduled Job automatically checks for warranty expiry and sends alerts, reducing unexpected downtime.

4. Real-Time Reporting

The system generates visual reports that help administrators understand asset distribution and utilization.

5. Reduced Manual Errors

Automation minimizes human mistakes in tracking and updating asset records.

6. Improved Operational Efficiency

Faster asset management processes save time and improve productivity.

7. Better Decision-Making

Reports and real-time data enable management to make informed asset-related decisions.

9. CONCLUSION

The Asset Management Portal developed using the ServiceNow platform provides a structured and automated solution for managing organizational assets throughout their lifecycle. By implementing a custom Asset Inventory table, UI Actions for status updates, a Scheduled Job for warranty expiry alerts, and real-time reporting, the system successfully improves asset tracking and operational efficiency.

The project demonstrates effective use of ServiceNow features such as automation, scripting, database management, and reporting tools. Through automated workflows and proactive maintenance alerts, the system minimizes asset downtime and reduces manual errors.

Overall, the Asset Management Portal enhances transparency, accountability, and decision-making capabilities within the organization, making asset management more reliable and efficient.

10. FUTURE ENHANCEMENTS

Although the current Asset Management Portal provides essential asset tracking and automation features, several enhancements can be implemented to further improve the system.

1. Role-Based Access Control

Implement different user roles such as Employee, Manager, and Asset Administrator to control access and permissions more effectively.

2. Service Portal Integration

Develop a user-friendly Service Portal interface where employees can request assets directly.

3. Approval Workflow Automation

Integrate Flow Designer to create automated approval workflows for asset requests.

4. Barcode / QR Code Integration

Enable barcode or QR code scanning for faster asset tracking and verification.

5. Email & SMS Notifications

Extend alert functionality to include SMS notifications in addition to email alerts.

6. Dashboard with Advanced Analytics

Create interactive dashboards with multiple reports for better asset performance analysis.

7. Integration with CMDB

Integrate the Asset Inventory table with Configuration Management Database (CMDB) for enterprise-level asset tracking.

8. Cloud Deployment & Scalability

Enhance system scalability for large organizations with high asset volumes.

11.Appendix

11.1 Source Code

SCRIPT :

```
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

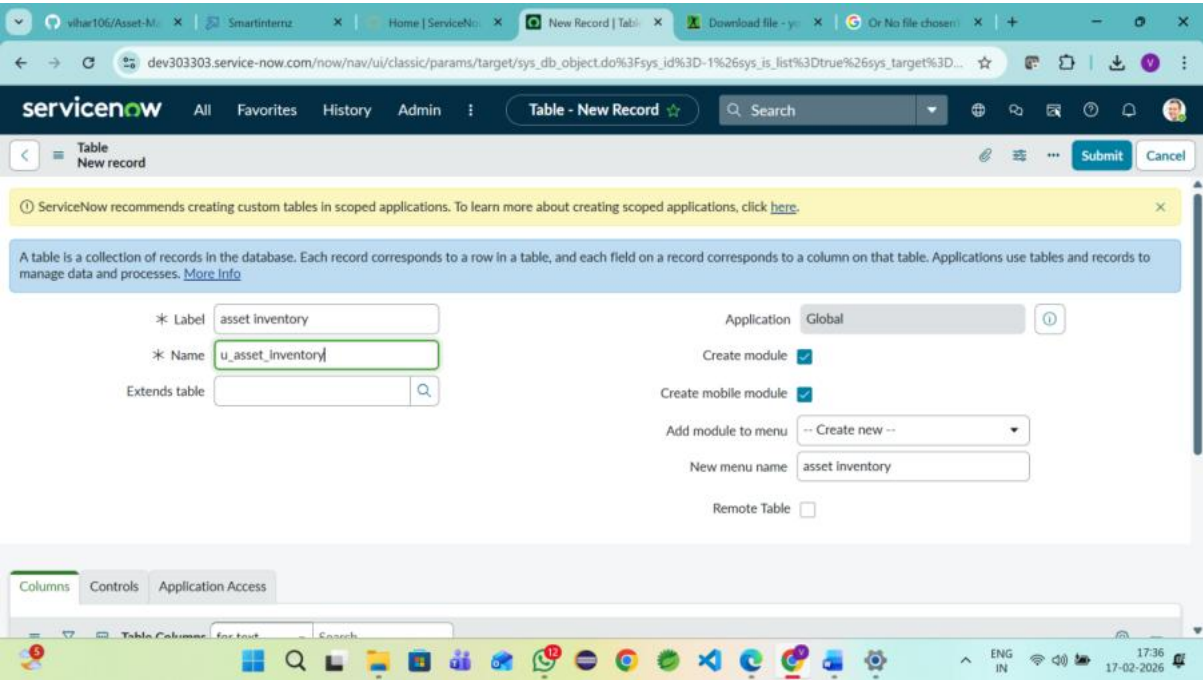
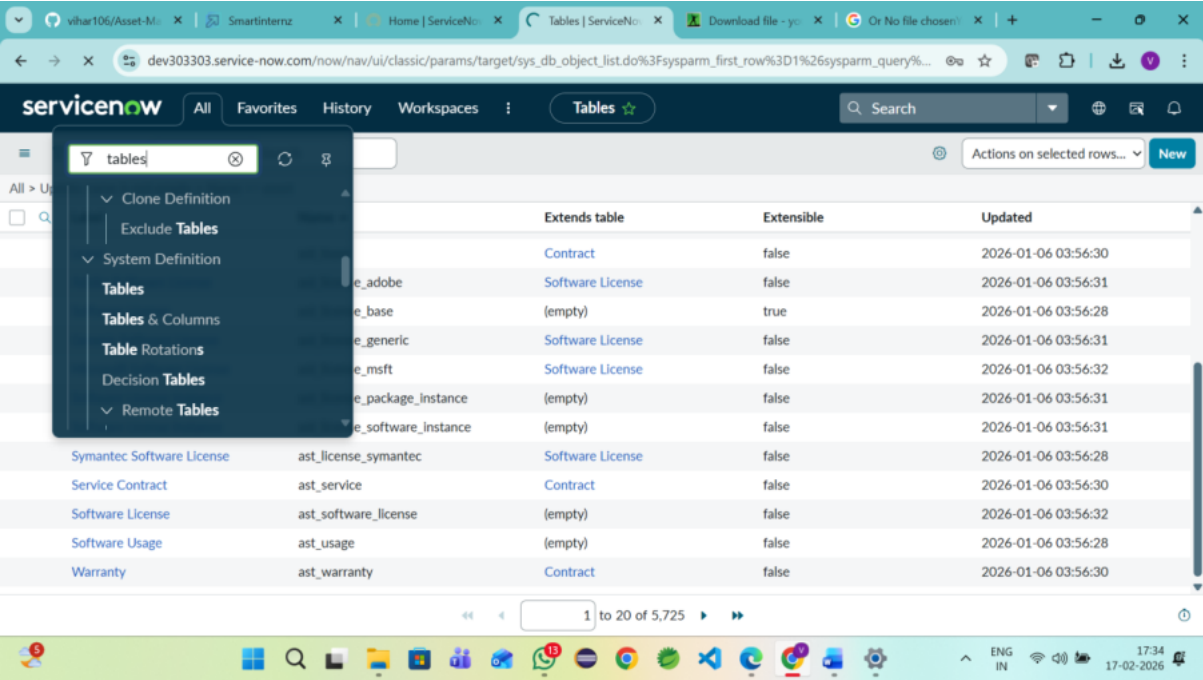
    email.send();

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

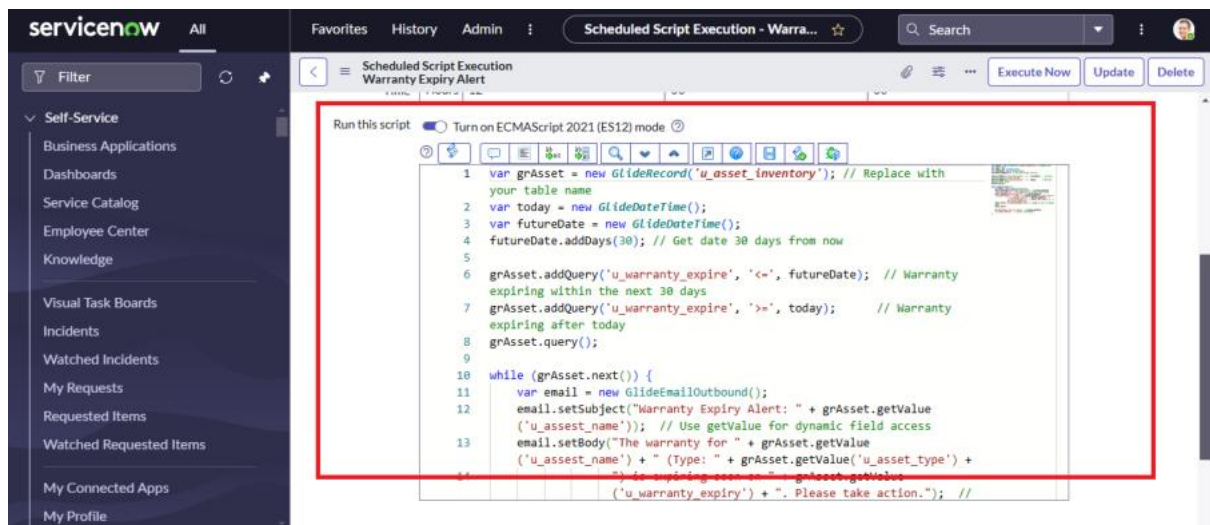
}
```

11.2 Screenshots

11.1 Asset Inventory Table



11.2 UI Action – Mark As Lost



11.3 Report – Available vs Assigned Assets

