

# **STREAMLINING TICKET ASSIGNMENT FOR EFFICIENT SUPPORT OPERATIONS**

# INTRODUCTION

Project Title: Rule-Based Ticket Assignment Automation System

Team Members: Ch .Kavya, G Siva Rama Krishna, Chalama

Reddy, Boggala Tirupathi Reddy

**Team ID :** LTVIP2026TMIDS24343

## PROJECT OVERVIEW

**Purpose:** The purpose of this initiative is to enhance the efficiency, accuracy, and responsiveness of ABC Corporation's support operations by minimizing manual intervention in ticket assignment. This will lead to improved customer satisfaction, better workload distribution, and optimal use of support team resources.

### Key Features:

- ❖ Automated Ticket Routing - Automatically assigns tickets based on predefined rules and categories.
- ❖ Priority-Based Assignment – Routes tickets based on urgency, impact, or SLA requirements.
- ❖ Team Skill Matching – Assigns tickets to agents or teams with the most relevant expertise.
- ❖ Real-Time Monitoring – Provides dashboards for live tracking of ticket status and workload.
- ❖ Rule Engine Configuration – Enables creation and modification of routing rules without coding.
- ❖ Integration with Support Tools – Seamlessly works with existing helpdesk or ITSM platforms.
- ❖ Load Balancing – Distributes tickets evenly to prevent overloading specific teams or agents.
- ❖ Audit Trail & Reporting – Maintains logs of routing decisions for transparency and analysis
- ❖ Scalability & Flexibility – Adapts easily to organizational growth or changes in support structure.

# Architecture

**Front-End (User Interface)** : The front-end deals with the user interface and interaction, ensuring that the data entered by users is accurate and optimized for automated processing

- ❖ Client Scripts

These are JavaScript scripts executed in the browser when users interact with the form (e.g., selecting a category).

- ❖ UI Policies

UI Policies dynamically control the visibility, read-only status, or requirement of fields on the form.

- ❖ Form Layouts & Variables

Custom fields and layouts guide users to provide the required information for ticket assignment.

**Back-End (Server-Side) Components** : The back-end is responsible for processing the ticket after submission and automatically assigning it using various logic and rules.

- Business Rules

Server-side scripts that trigger before or after record insert/update.

- Assignment Rules

Built-in ServiceNow feature that automatically assigns ticket based condition like category, priority, or caller.

- Flow Designer / Workflows

No-code/low-code automation tools to route tickets, send notifications, or trigger approvals.

- Script Includes

Custom reusable functions written in JavaScript for more complex assignment logic.

## Database

The database stores all the ticket and assignment data. ServiceNow is built on a relational database model, where every ticket, group, and user is a record in a Table.

**Key Tables Involved:**

incident → Stores incident records (tickets).

sys\_user → Stores user details (callers, agents).

sys\_user\_group → Stores assignment groups.

cmdb\_ci → Stores configuration items, useful for impact-based assignment.

task (parent table) → Generic table for all task types (incident, change, etc.)

## Setup instructions

- ❖ Set up clear assignment rules based on ticket fields like category, priority, or location.
- ❖ Create assignment groups and assign users to the right teams.
- ❖ Automate routing using Business Rules or Flow Designer for consistent and fast assignment.

## Folder structure

```
Streamlined-Ticket-Assignment-ServiceNow/
└── API_Integration/
    ├── API_README.md
    ├── create_incident.py
    ├── get_incident_by_id.py
    ├── update_assignment.py
    └── delete_ticket.py
    └── API_Documentation.pdf
```

## API Documentation

### User:

```
import requests
from requests.auth import HTTPBasicAuth
url = "https://dev270972.service-now.com/$m.do#/home"
auth = HTTPBasicAuth("admin", "password")
payload = {
    "first_name": "Manne",
    "last_name": "Niranjan",
    "email": "niranjanreddymanne2507@gmail.com",
    "user_name": "manne.niranjan",
    "password": "niranjan123",
}

headers = {"Content-Type": "application/json"}

res = requests.post(url, auth=auth, json=payload, headers=headers)
print(res.json())
```

### Project:

```
url = "https://dev270972.service-now.com/$m.do#/home"
payload = {
```

```
        "short_description": "New College Project Support",
        "description": "Need technical support for Java assignment project",
        "category": "project",
        "assignment_group": "Project Support"
    }
res = requests.post(url, auth=auth, json=payload, headers=headers)
print(res.json())
```

## Application Issue:

```
payload = {
    "short_description": "Application not responding",
    "description": "App crashes during launch",
    "category": "application",
    "assignment_group": "Application Team",
    "priority": "2"
}

res = requests.post(url, auth=auth, json=payload, headers=headers)
print(res.json())
```

## Chat:

```
chat_message = "Hi team, my laptop is overheating during calls."
payload = {
    "short_description": "Chat Submission: Laptop Heating",
    "description": chat_message,
    "category": "hardware",
    "assignment_group": "Hardware Team"
}

res = requests.post(url, auth=auth, json=payload, headers=headers)
print(res.json())
```

## Assign Ticket:

```
def auto_assign(category):
    group = {
        "software": "Software Team",
        "hardware": "Hardware Team",
        "network": "Network Team",
        "project": "Project Support"
    }.get(category, "IT Helpdesk")

    payload = {
        "short_description": f"{category} ticket via API",
        "category": category
    }

    res = requests.post(url, auth=auth, json=payload, headers=headers)
    print(res.json())
```

```
        "category": category,
        "assignment_group": group
    }
res = requests.post(url, auth=auth, json=payload, headers=headers)
print(res.json())

auto_assign("software")
```

### Ticket status:

```
ticket_sys_id = "abcd1234567890"
url = f
"https://your_instance.service-now.com/api/now/table/incident/{ticket_sys_id}"
res = requests.get(url, auth=auth, headers=headers) print(res.json()['result']
['state'])
```

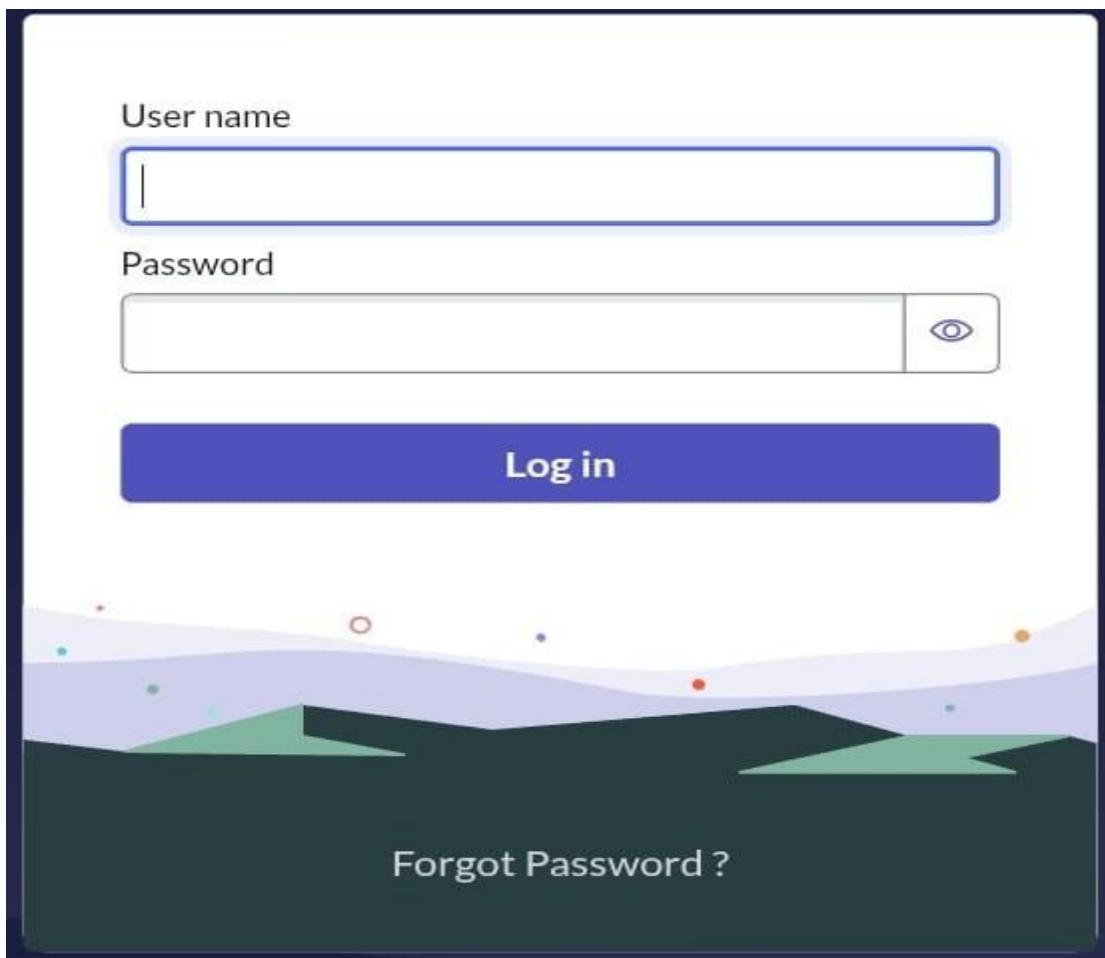
## Authentication Flow

- ❖ REST APIs to create or assign incidents
- ❖ External integrations (chatbots, portals, mobile apps)
- ❖ Scripted APIs to automate ticket routing

## Testing

- ❖ Tested ticket creation via UI, Python scripts, and Postman (API).
- ❖ Checked Client Scripts and UI Policies for field visibility and resets.
- ❖ Validated Business Rules and Assignment Rules on record creation.
- ❖ Performed negative testing with blank/invalid inputs.
- ❖ Ensured secure access using Basic Auth and role-based permissions.
- ❖ Confirmed expected outputs via response codes and assignment logs.

## Screenshots/Demo



A screenshot of the ServiceNow user profile edit screen for Katherine Pierce. The top navigation bar shows tabs for "All", "Favorites", "History", and the current "User - Katherine Pierce". The main content area displays various user profile fields: User ID (Katherine Pierce), First name (Katherine), Last name (Pierce), Title (T), Department (dropdown menu), Password needs reset (checkbox), Locked out (checkbox), Active (checkbox checked), Web service access only (checkbox), Internal Integration User (checkbox), Email (dropdown menu), Language (dropdown menu set to "None"), Calendar integration (dropdown menu set to "Outlook"), Time zone (dropdown menu set to "System (America/Los\_Angeles)"), Date format (dropdown menu set to "System (yyyy-MM-dd)"), Business phone (text input), Mobile phone (text input), and Photo (link to "Click to add..."). At the bottom of the screen are buttons for "Update", "Set Password", and "Delete".

Servicenow

User - Manne Nirjanan

User ID: manne.niranjan

First name: Manne

Last name: Nirjanan

Title:

Department:

Password needs reset:

Locked out:

Active:

Web service access only:

Internal Integration User:

Email: niranjanreddymanne2507@gmail.com

Language: None

Calendar integration: Outlook

Time zone: System (America/Los\_Angeles)

Date format: System (yyyy-MM-dd)

Business phone:

Mobile phone:

Photo: Click to add...

Update Set Password Delete

Related Links

30°C Mostly cloudy

Search web &... hp

Servicenow

Group - Platform

Name: Platform

Manager: Manne Nirjanan

Description:

Group email:

Parent:

Update Delete

Roles (1) Group Members (1) Groups

Role: Platform\_role

Created: 2025-06-21 10:42:02

Granted by: (empty)

Inherits: true

Actions on selected rows...

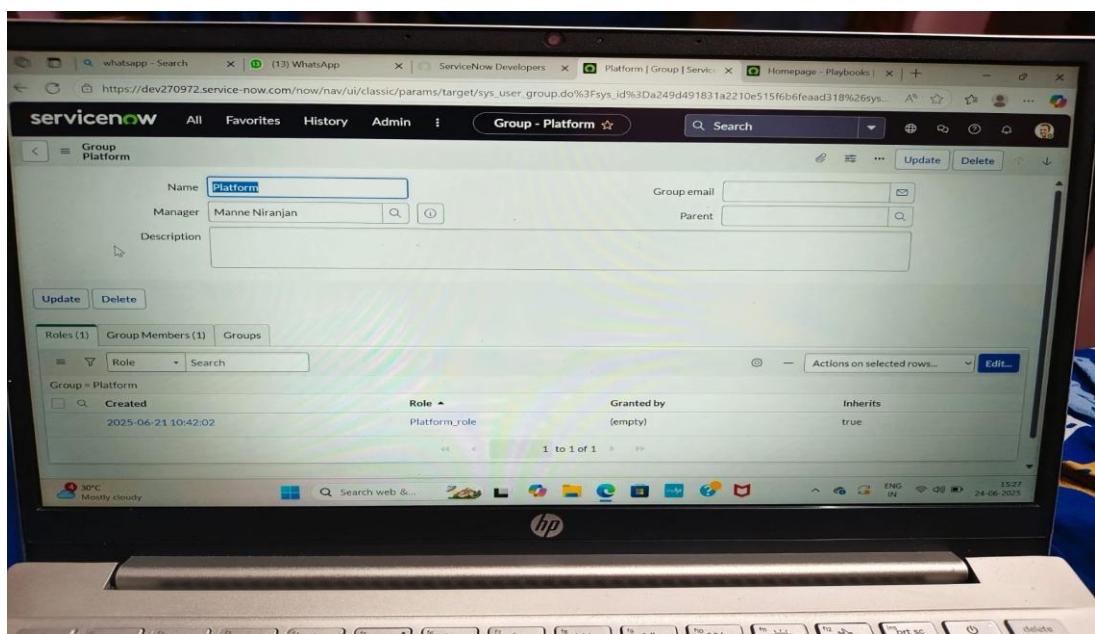
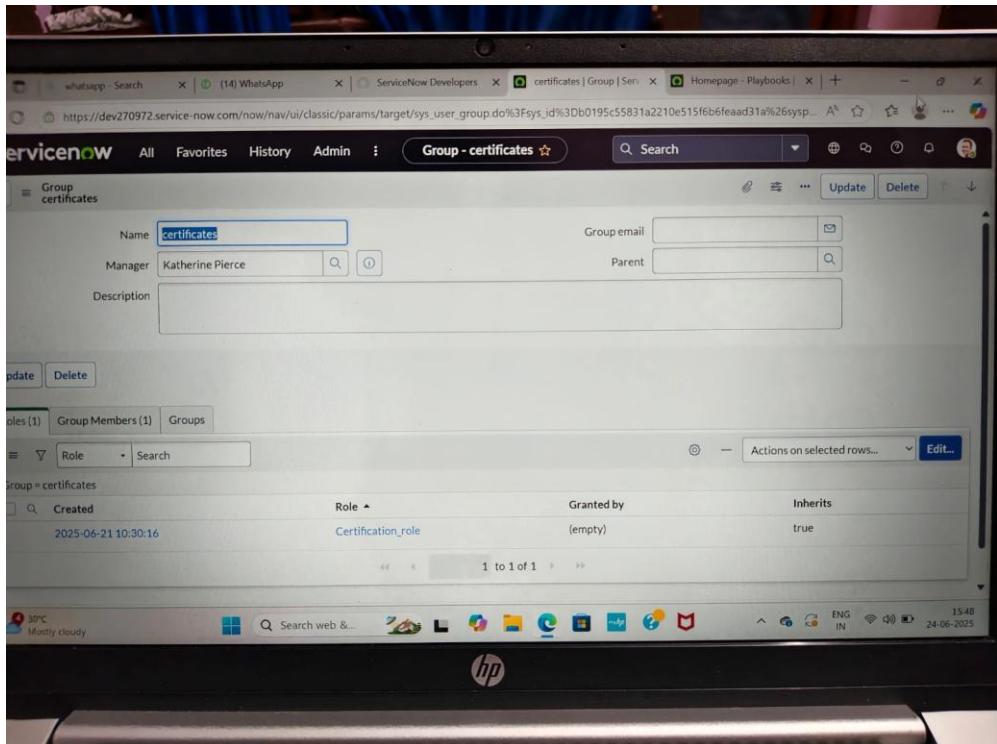
30°C Mostly cloudy

Search web &... hp

The screenshot shows the ServiceNow Role - Platform\_role configuration page. The role is named "Platform\_role" and is set to "Global". It has a description: "Can deal with platform related issues". The "Elevated privilege" checkbox is unchecked. Below the role details, there are "Related Links" and a search bar for "Contains Roles". The search results show one entry: "Role = Platform\_role". The operating system taskbar at the bottom indicates it's an HP laptop with a weather widget showing "30°C Mostly cloudy".

The screenshot shows the ServiceNow Table - Operations related list view. The table lists various fields and their properties. The columns include Sys ID, Type, Label, Value, and Script. The table has 15 rows, each corresponding to a field like Sys ID, Updates, Priority, etc. The operating system taskbar at the bottom indicates it's an HP laptop with a weather widget showing "Light rain At night".

Table - Operations related					
Operations related	Sys ID	Type	Label	Value	Script
	Sys ID	Sys ID (GUID)	(empty)	32	false
	Updates	Integer	(empty)	40	false
	Priority	String	(empty)	40	false
	Created	Date/Time	(empty)	40	false
	Updated by	String	(empty)	40	false
	Assigned to group	Reference	Group	32	false
	Updates	Integer	(empty)	40	false
	Issue	String	(empty)	40	false
	Class	System Class Name	(empty)	80	javascript:current.getTableName();
	Ticket raised Date	Date/Time	(empty)	40	false
	Name	String	(empty)	40	false
	Comment	String	(empty)	40	false
	Updated	Date/Time	(empty)	40	false
	Updated	Date/Time	(empty)	40	false
	Updated by	String	(empty)	40	false
	Created	Date/Time	(empty)	40	false



servicenow Access Control - u\_operations\_related\_u\_priority

Access Control Rules have two decision types, and these types will behave differently depending on conditions.

1. Allow Access: Allows access to a resource if all conditions are met.  
2. Deny Access: Denies access to a resource unless all conditions are met.

More Info

Requires role

Role

admin

Insert a new row...

Security Attribute Condition

Local or Existing  Existing  Local

Condition All of these conditions must be met

-- choose field --

OR AND

New Criteria

servicenow Access Controls

Access Control - u\_operations\_related\_u\_issue

Type: record  
Operation: write  
Decision Type: Allow If

Admin overrides:

Protection policy: None

Name: issue [u\_operations\_related\_u\_issue]

Description: Default access control on u\_operations\_related\_u\_issue

Applies To: Add Filter Condition Add "OR" Clause

servicenow Access Control - u\_operations\_related\_u\_name

Type: record  
Operation: write

Admin overrides:

Protection policy: None

Name: name [u\_operations\_related\_u\_name]

Description: Default access control on u\_operations\_related\_u\_name

Conditions

Access Control Rules have two decision types, and these types will behave differently depending on conditions.

1. Allow Access: Allows access to a resource if all conditions are met.  
2. Deny Access: Denies access to a resource unless all conditions are met.

More Info

servicenow Access Control - u\_operations\_related.u\_service\_request\_no

Type: record  
Operation: write

Admin overrides:

Protection policy: None

Name: Operations related [u\_operations\_related]  
Service request No

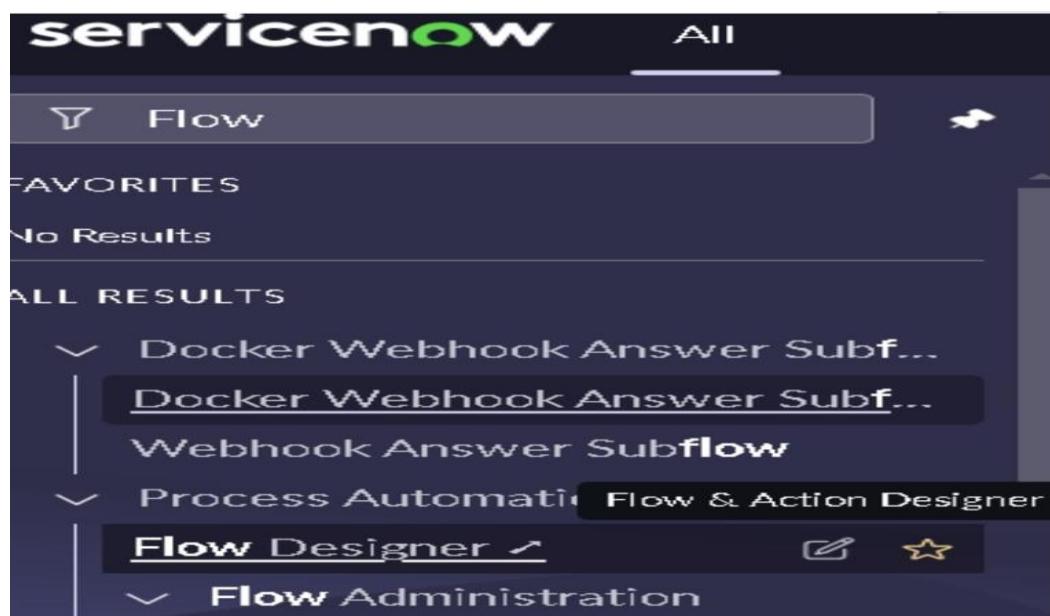
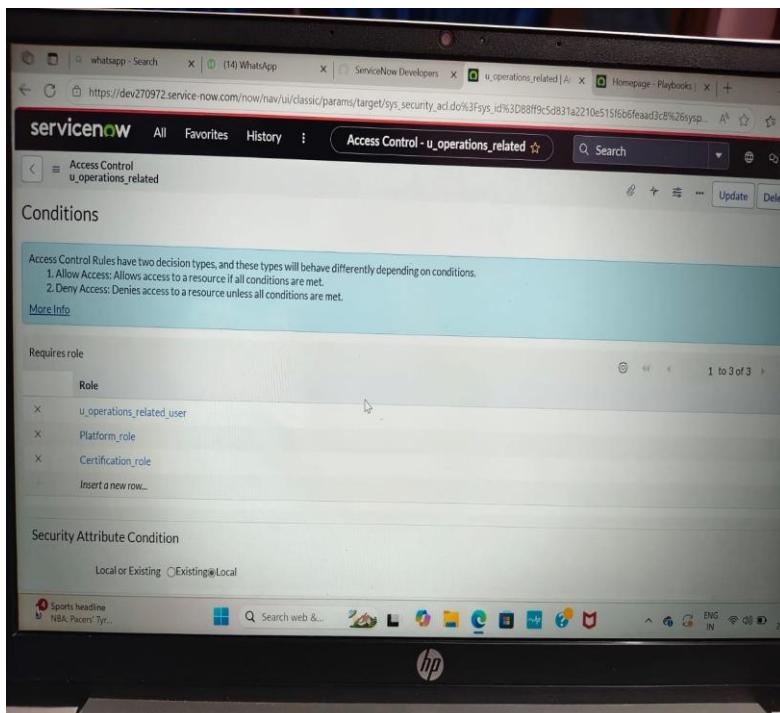
Description:

Conditions

Access Control Rules have two decision types, and these types will behave differently depending on conditions.

1. Allow Access: Allows access to a resource if all conditions are met.  
2. Deny Access: Denies access to a resource unless all conditions are met.

More Info



### Flow properties

Flow name

Description

Application

Protection

Run As

[Cancel](#) [Submit](#)

## Let's get the details for your flow

Flow name \* ⓘ

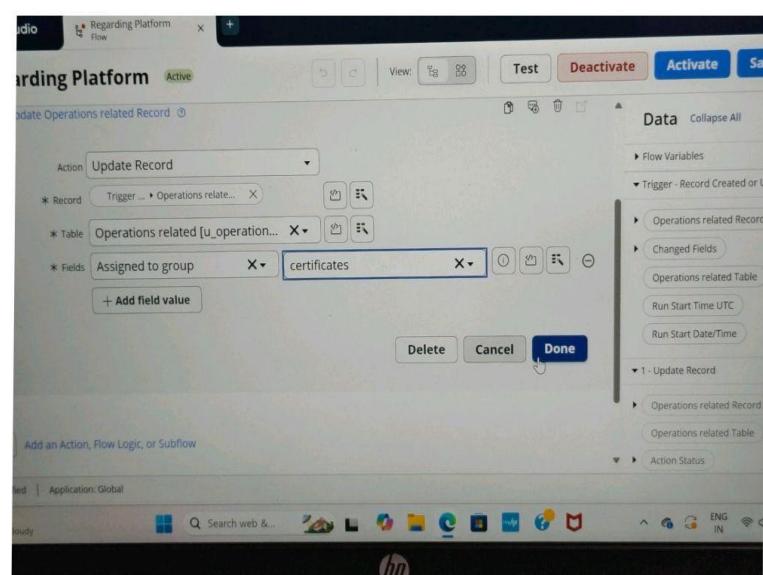
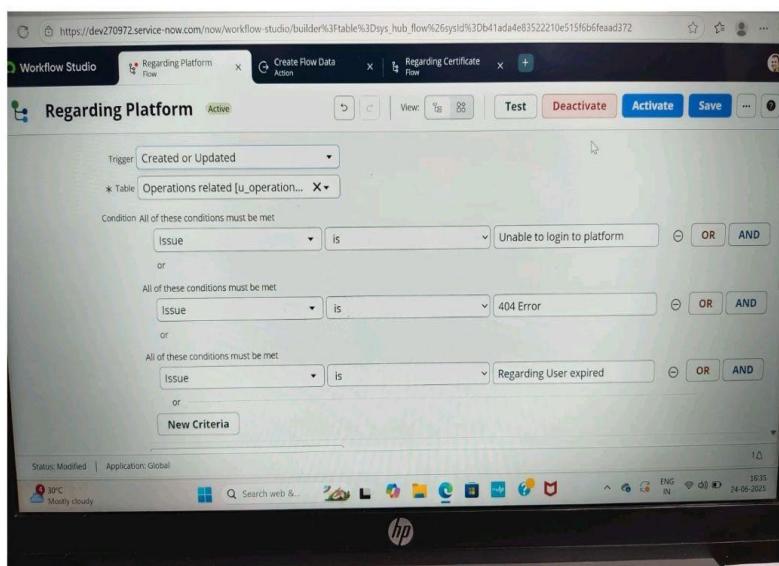
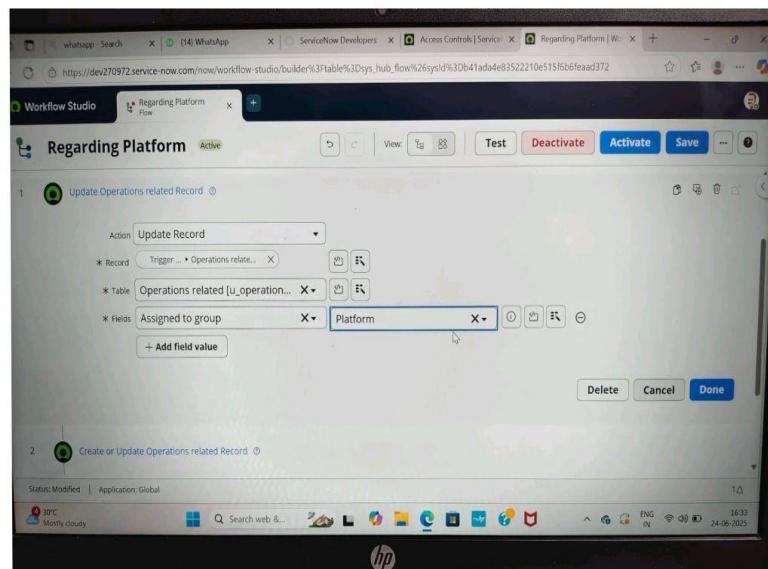
Application \* ⓘ

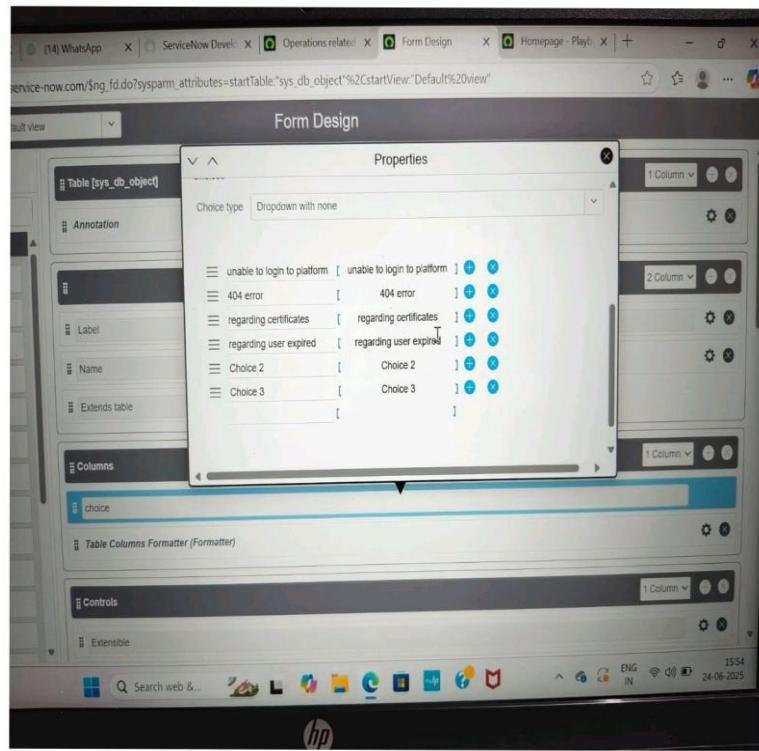
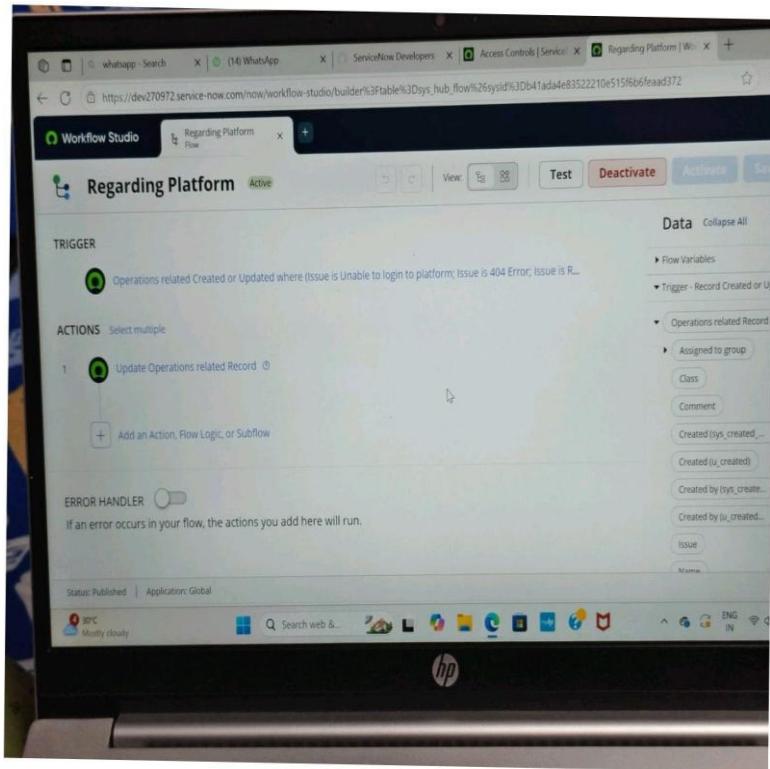
Description ⓘ

Hide additional properties

Protection ⓘ  
 Option that you can select to specify that the flow runs as a system user or the user who initiates the session.

Run as ⓘ





## Known Issues

- ❖ Hardcoded Assignment Mapping  
Assignment groups are mapped based on static category values; dynamic or custom categories may not route correctly.
- ❖ Basic Authentication in Scripts  
Current API programs use Basic Auth, which is not secure for production use without HTTPS or token-based auth.
- ❖ No Real-Time Chat Integration  
Chat submissions are simulated; real-time integration with Virtual Agent or MS Teams is not yet implemented.

## Future Enhancements

- ❖ Integrate AI-based Assignment  
Use ServiceNow Predictive Intelligence to auto-assign tickets based on historical patterns and agent performance.
- ❖ Switch to OAuth 2.0 Authentication  
Replace Basic Auth in APIs with secure, token-based OAuth for better security in production environments.
- ❖ Real-Time Chatbot Integration  
Connect with Virtual Agent, MS Teams, or WhatsApp to allow real-time ticket submission and updates via chat.
- ❖ Custom Assignment Rules for Location & Priority  
Enhance assignment logic to include location, impact, and urgency for more accurate ticket routing.
- ❖ Add Retry and Logging in API Scripts  
Improve API resilience by adding retry logic, logging failed requests, and sending alerts on failure.