**LAPTOP REQUEST CATALOG ITEM**

**Team ID:NM2025TMID14896**

**Team members:5**

**Team Leader : MONISHA A(monishaanada@gmail.com)**

**Team member 1:NANDHAKUMAR K (nandhakrn1@gmail.com)**

**Team member 2:PRAVEEN S(praveenpavi1494@gmail.com)**

**Team member 3:VISHAL V (vishalvishal58757@gmail.com)**

**Team member 4:YUVARAJ C (devayuva61@gmail.com)**

**Problem Statement**

Employees need a fast, reliable way to request laptops. The existing process is manual, slow, and error-prone: forms aren’t dynamic to guide users, required information is often missed, there’s no simple reset option, and changes aren’t tracked for controlled deployment.

**Objective**

To streamline and automate the laptop request process by creating a dynamic, user-friendly Service Catalog item in ServiceNow.

To eliminate manual, error-prone methods by implementing guided forms, conditional logic (UI policies), and reset functionality.

To ensure all configuration changes are properly tracked and transferable between environments through update sets.

To improve efficiency, accuracy, and governance in laptop provisioning requests.

**Skills Demonstrated**

**1. ServiceNow Configuration**

Creating and managing Service Catalog Items.

Designing and adding variables (form fields) for structured data collection.

**2. UI Policies & Policy Actions**

Implementing dynamic form behavior (show/hide fields, mandatory conditions).

**3. UI Actions (Client-side Scripting)**

Writing client-side scripts for enhanced usability (e.g., reset form button).

**4. Update Set Management**

Capturing, exporting, importing, and committing update sets for controlled deployment.

**5. Testing & Validation**

End-to-end testing of catalog items and workflows to ensure correct behavior and user experience.

**6. Problem-Solving & Process Automation**

Analyzing manual processes and transforming them into efficient, automated solutions.

**Update set**

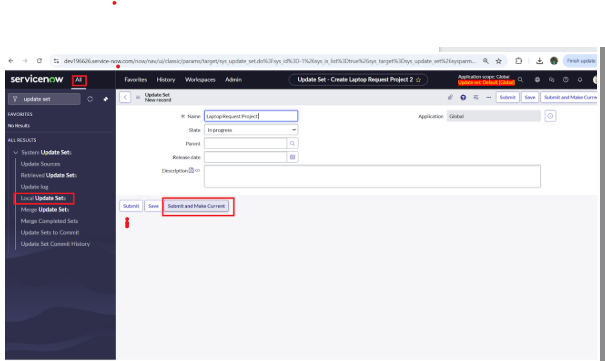
**Step-by-step process**

**Create a local update set (capture all changes)**

1. All → System Update Sets → Local Update Sets → New.

2. Name: Laptop Request (or Laptop Request Project). Add description if needed.

3. Click Submit, then click Make Current.

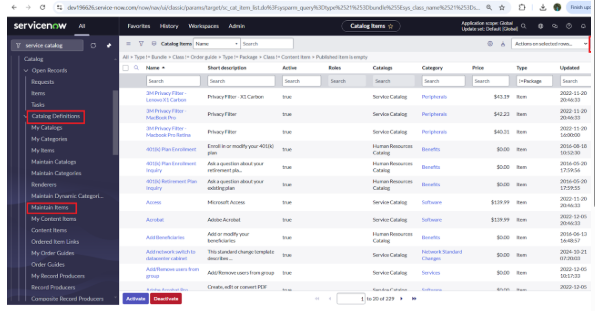


> Perform every subsequent action while this update set is current so all changes are recorded.

**Service Catalog Item**

**Create the Service Catalog item**

1. All → Service Catalog → Catalog Definitions → Maintain Items → New.



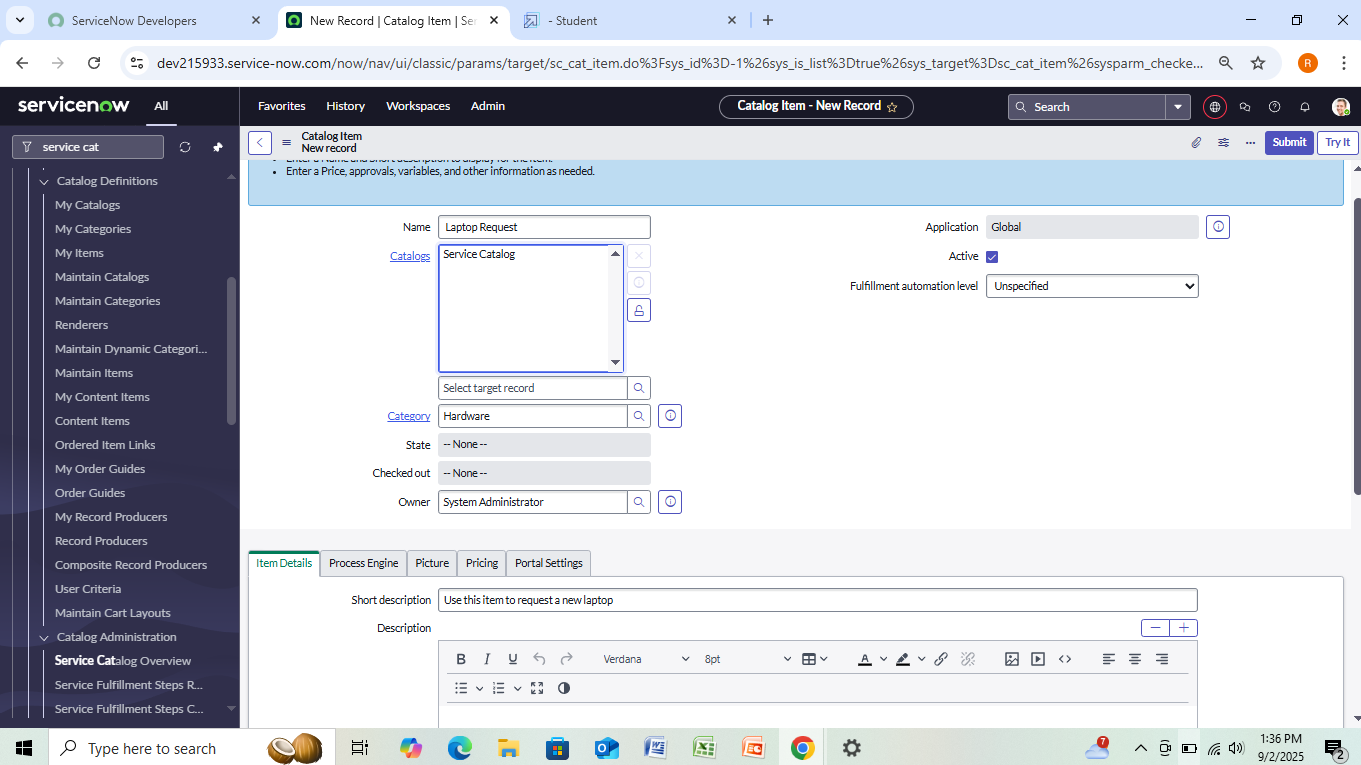
2. Fill in:

Name: Laptop Request

Catalog: Service Catalog

Category: Hardware

Short description: Use this item to request a new laptop



3. Click Save.

**Add variables (form fields)**

On the saved catalog item, use the Variables related list → New for each variable:

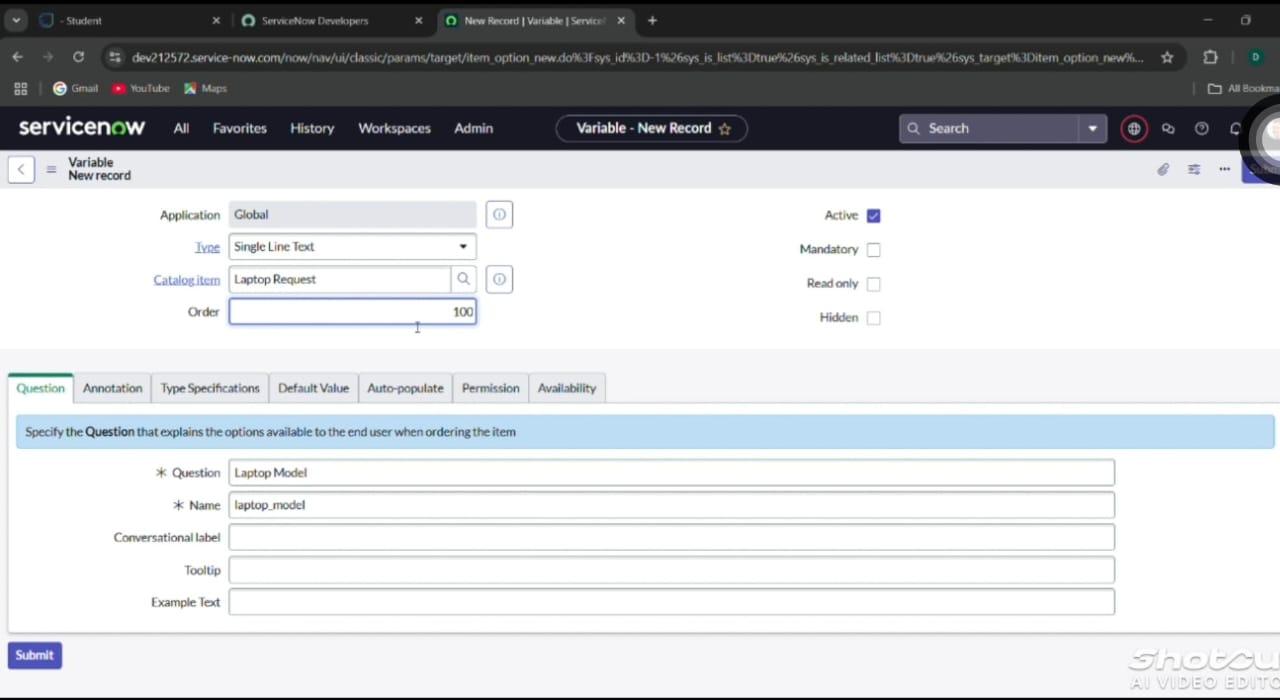
**Variable 1**

Label: Laptop Model

Type: Single line text

Name: laptop\_model

Order: 100 → Submit



**Variable 2**

Label: Justification

Type: Multi-line text

Name: justification

Order: 200 → Submit

**Variable 3**

Label: Additional Accessories

Type: Checkbox

Name: additional\_accessories

Order: 300 → Submit

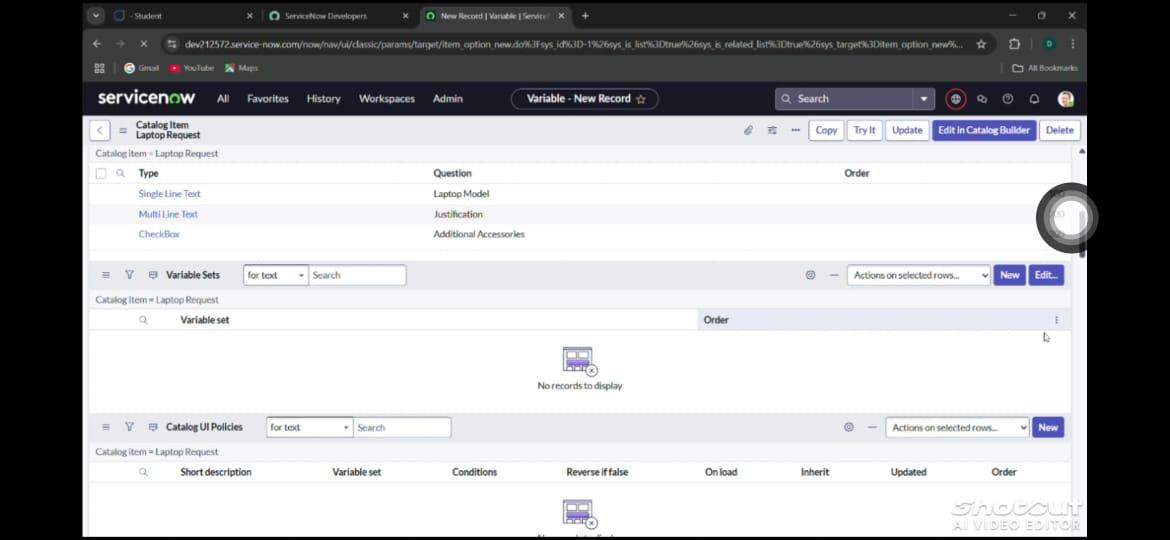
**Variable 4**

Label: Accessories Details

Type: Multi-line text

Name: accessories\_details

Order: 400 → Submit



After adding variables, Save the catalog item form.

**UI Policy**

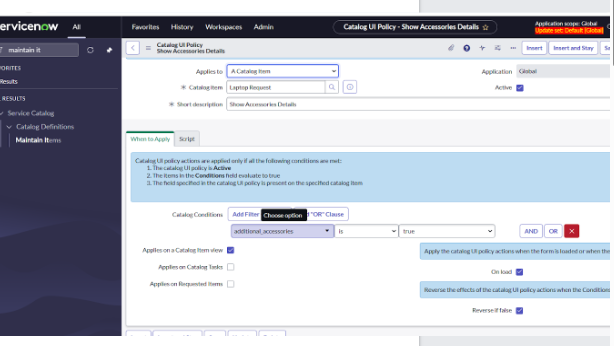
**Create Catalog UI Policy (show + mandatory behavior)**

1. On the catalog item form, scroll to Catalog UI Policies → New.

2. Short description: Show accessories details.

3. When to apply (Catalog Condition): set Field additional\_accessories is true.

4. Click Save (do not click Submit if your process prefers Save).



5. In the same policy, open Catalog UI Policy Actions → New and set:

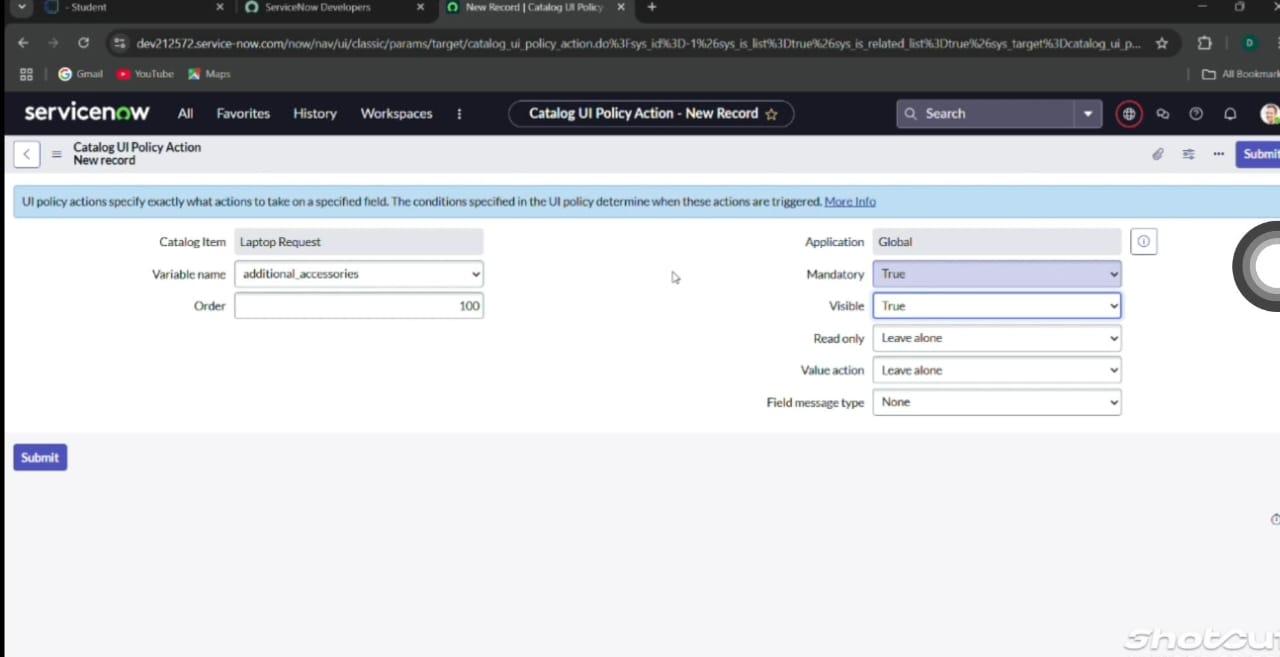
Variable name: accessories\_details

Order: 100

Mandatory: True

Visible: True → Save

6. Click Save again on the Catalog UI Policy record.



**Result:**

when the user checks Additional Accessories, Accessories Details becomes visible and mandatory.

**UI Action**

**Create UI Action (Reset form button)**

1. All → System Definition → UI Actions → New.

2. Fill:

Table: sc\_cart (or sc\_cart\_item if you want it on the item)

Order: 100

Action name / Label: Reset form

Check Client (so it runs client-side).

3. Script (client-side):

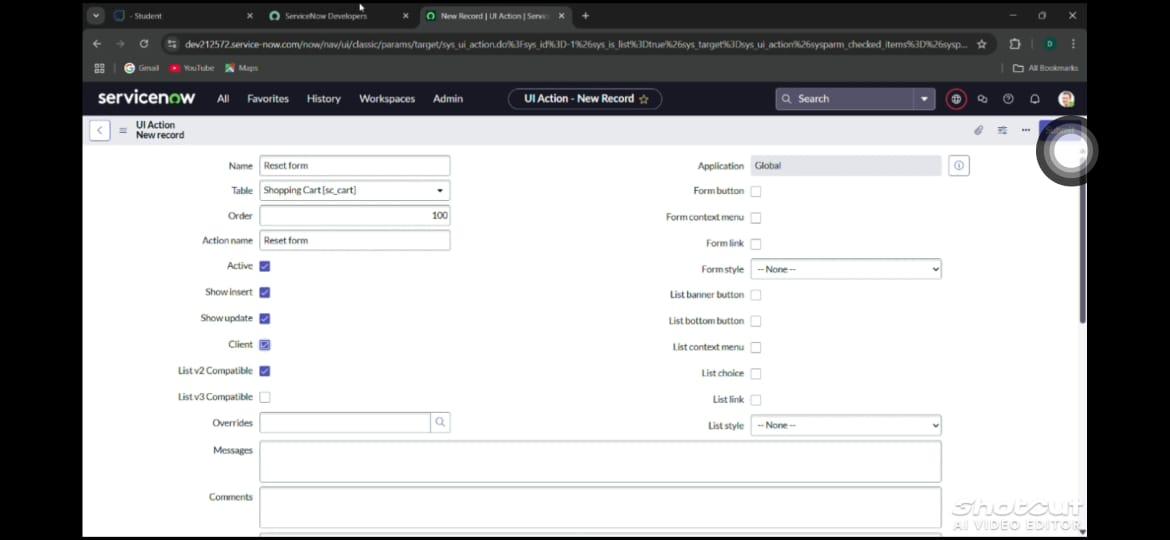
function resetForm() {

g\_form.clearForm(); // Clears all fields in the form

alert("The form has been reset.");

}

4. Click Save.



**Export Update Set**

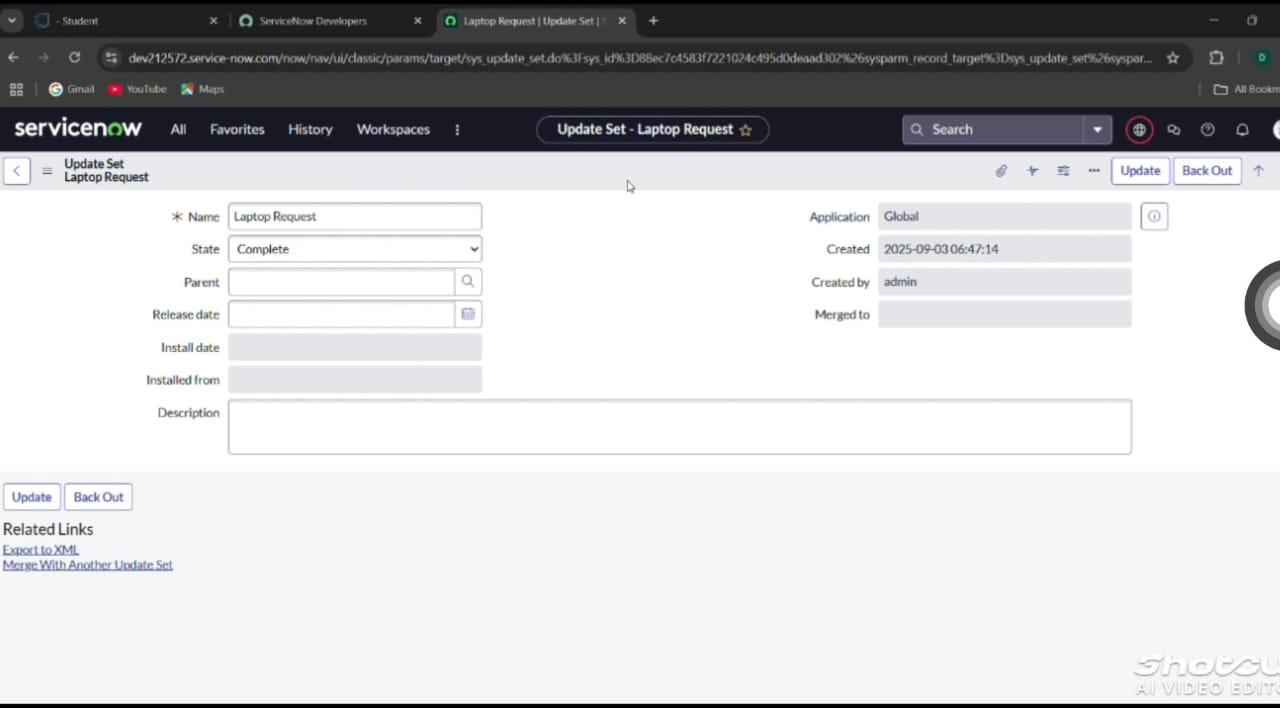
**Complete and export the update set**

1. All → System Update Sets → Local Update Sets → open Laptop Request.

2. Change State to Complete.

3. Verify the Updates related list contains the catalog item, variables, UI policy, UI policy action, and UI action.

4. Click Export to XML and download the XML file.

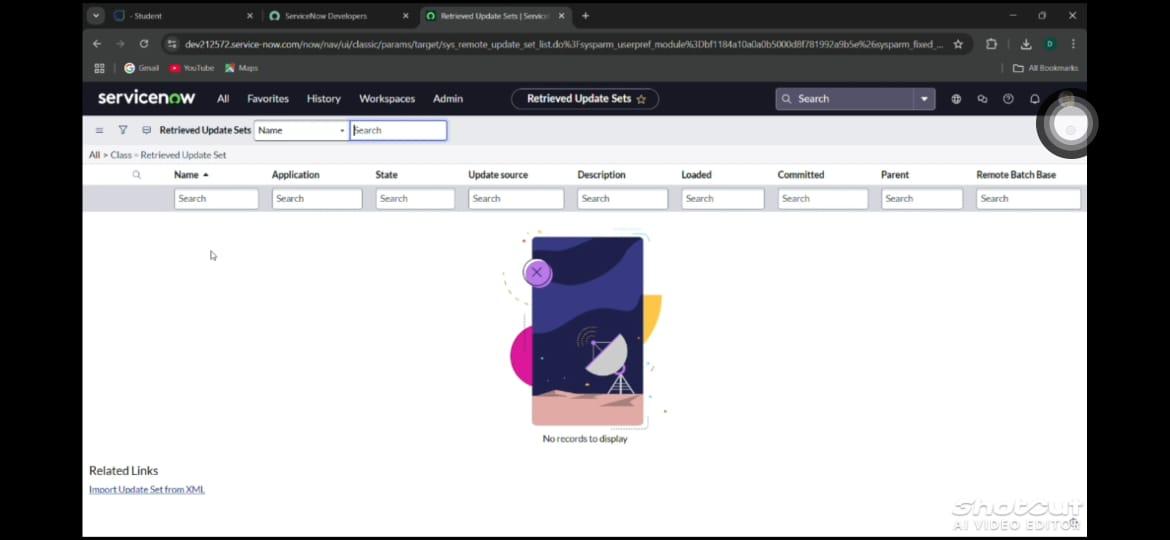


**Login To Another Instance**

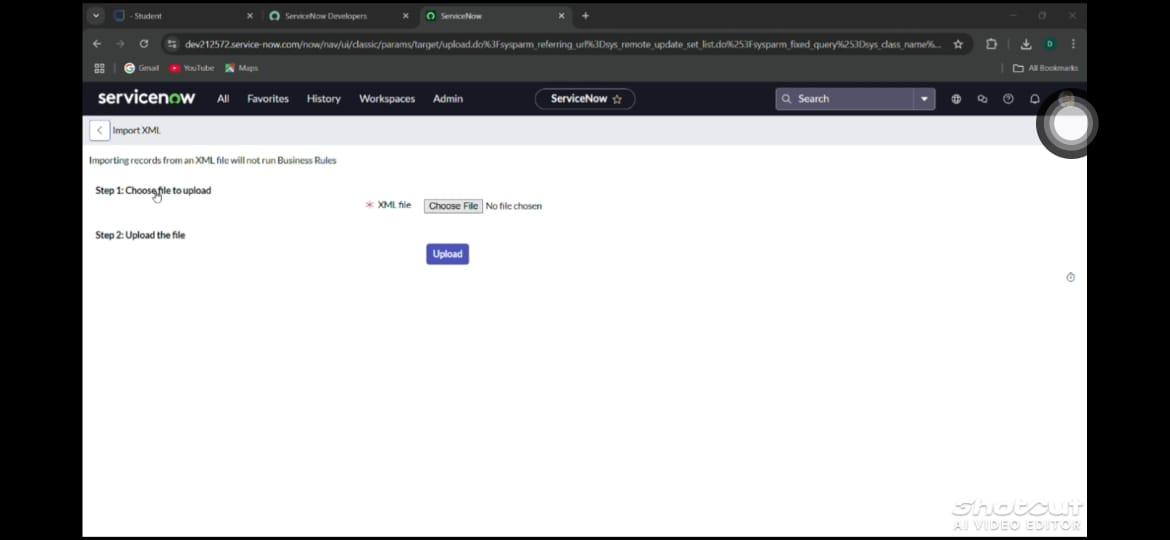
**Import & commit update set in target instance**

1. Log into target instance. All → System Update Sets → Retrieved Update Sets.

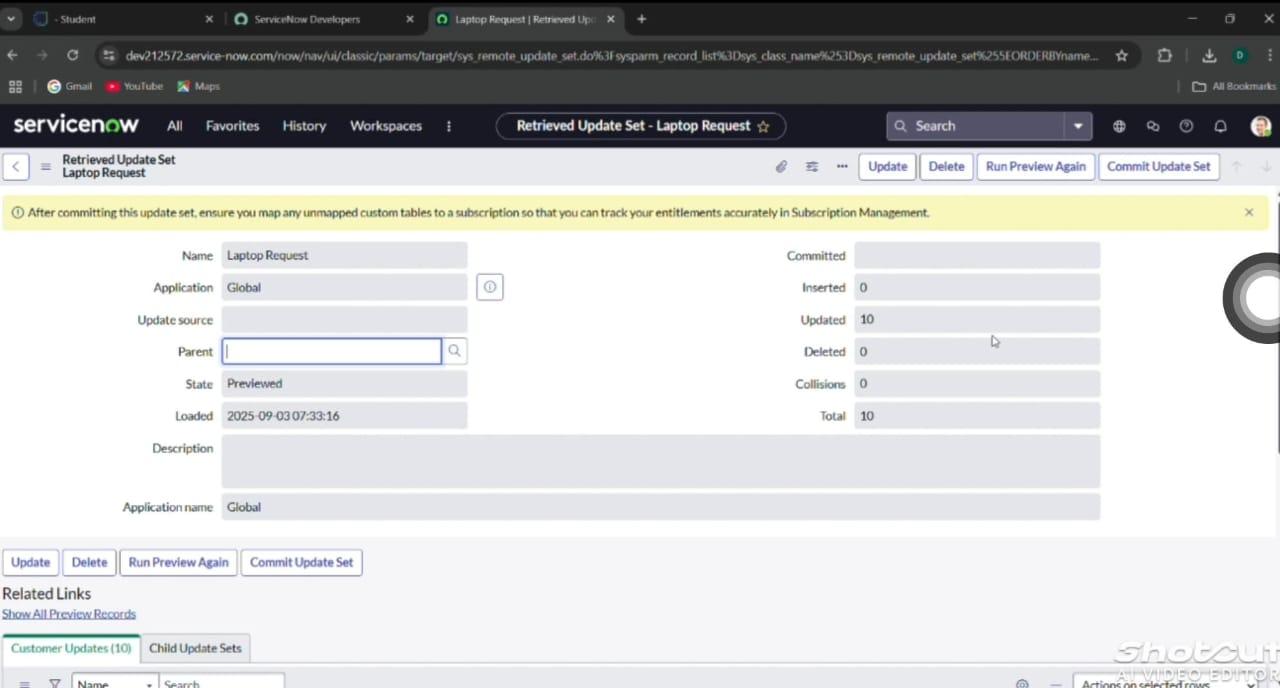
2. Click Import Update Set from XML, upload the XML file → Upload.



3. Open the retrieved update set → Preview Update Set (resolve any preview issues).



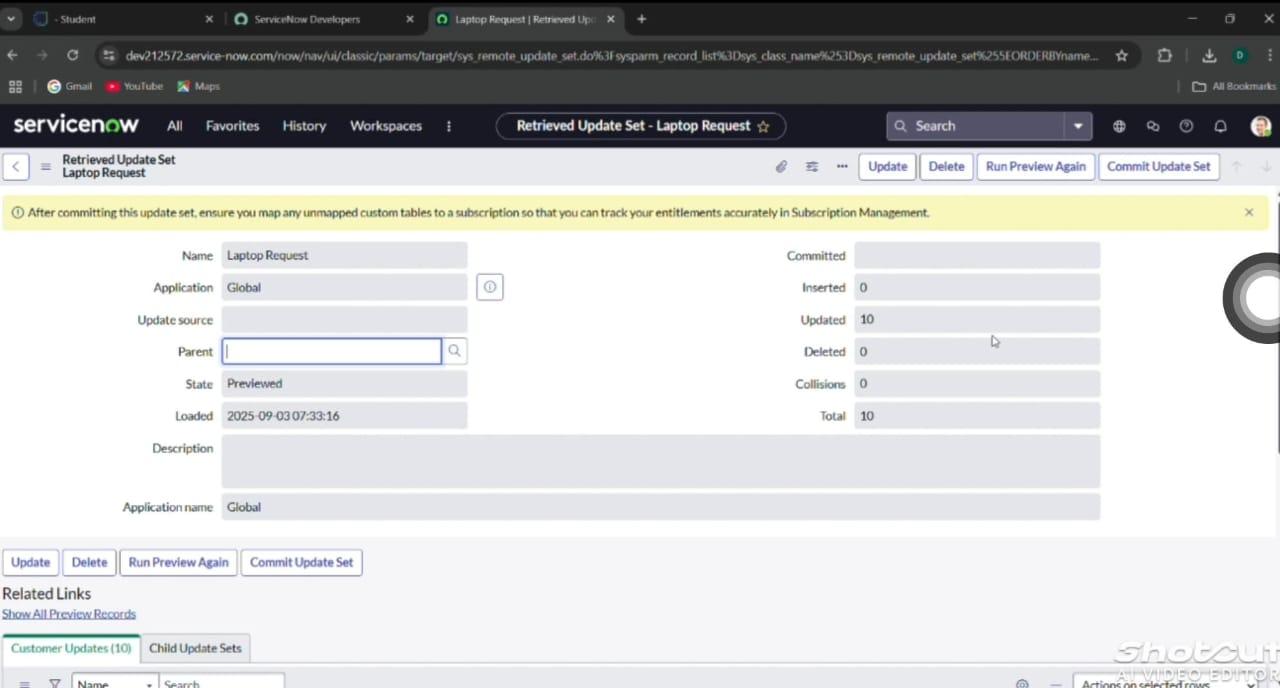
4. Click Commit Update Set. Confirm the Updates related list shows imported objects.



**Testing**

**Test the catalog item (validation)**

1. In target instance: Service Catalog → Catalogs → Service Catalog (or Hardware) → open Laptop Request.



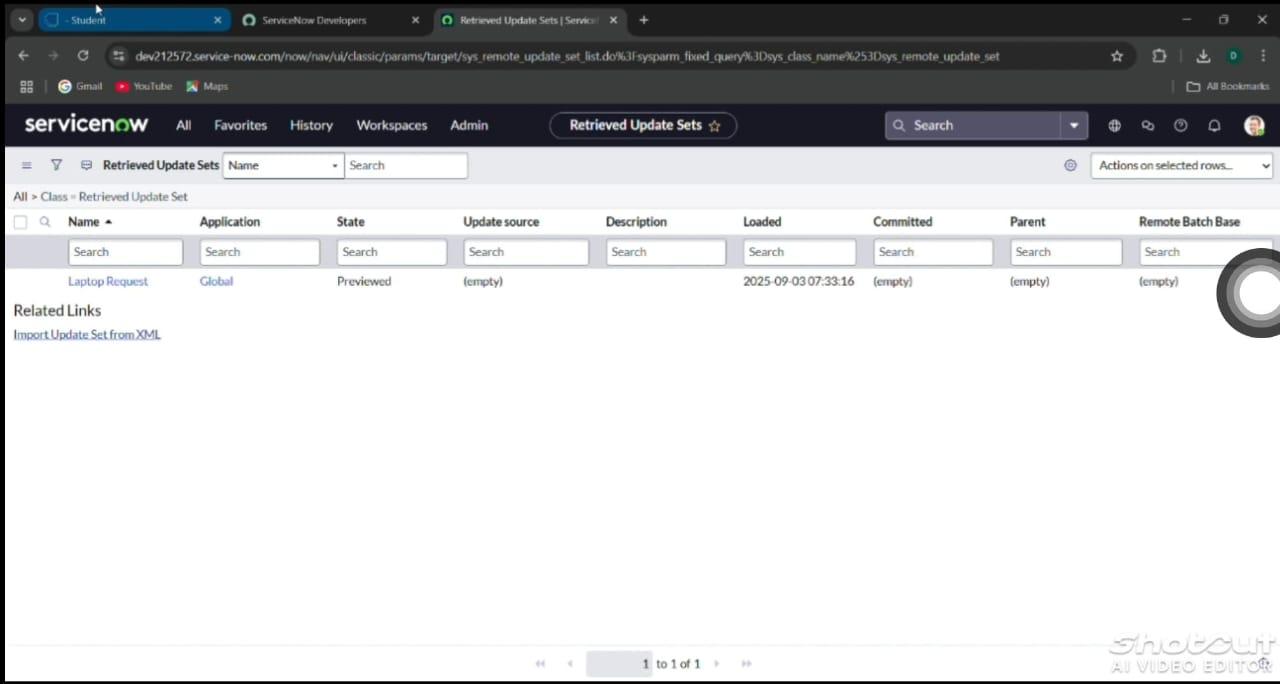
2. Verify fields: Laptop Model, Justification, Additional Accessories, Accessories Details (hidden initially).

3. Test behavior:

Leave Additional Accessories unchecked → accessories\_details stays hidden.

Check Additional Accessories → accessories\_details appears and is mandatory.

4. Add the item to cart and verify Reset form works (clears fields) in the cart or item area where created.



**Conclusion**

Creating the Laptop Request ServiceNow catalog item replaces a slow, manual laptop request process with a tracked, dynamic, user-friendly form. Dynamic fields and UI policy logic ensure required details are captured only when relevant, the reset button improves usability, and using an update set guarantees changes are governed and portable between instances — resulting in faster requests, fewer errors, and better governance.