Syncify - Functionality Guide

Overview:

- This document outlines the core functional aspects of the Syncify App within Salesforce.
- It details key features like Real-Time and Bi-Directional data synchronization between Shopify and Salesforce, allowing seamless updates for objects such as Accounts, Contacts, Products and Orders etc.
- The document also covers customizable field mapping, enabling users to define how Shopify and Salesforce fields align.
- Additionally, it explains the batch import process for Shopify data into Salesforce, allowing efficient initial syncing of records.
- The app triggers real-time updates for create, update, and delete events in Shopify, ensuring data consistency across platforms.
- These functionalities simplify data management and enhance operational workflows between the two systems.

Initial Sync Overview and Setup:

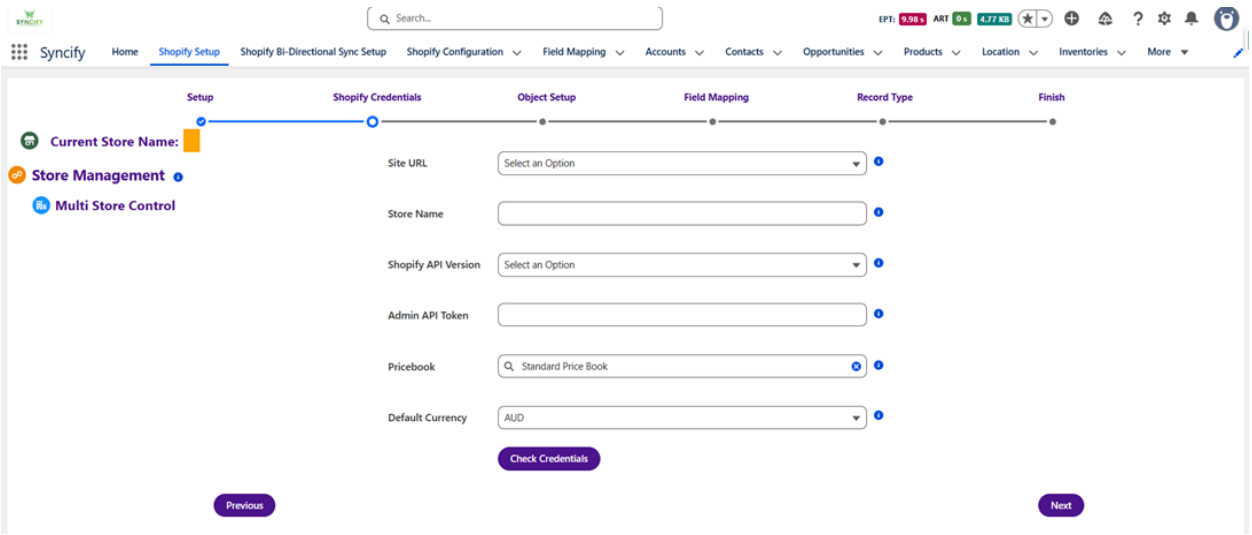
Initially, in the Syncify App, the following pages are sequentially aligned to provide users with a streamlined, step-by-step process for configuring a store and enabling flawless data synchronization:

- **Setup** – Initiates the configuration process, guiding users to set up the foundational elements for the Syncify App.
- **Shopify Credentials** – Allows users to enter and validate their Shopify store credentials, ensuring a secure connection between Shopify and Salesforce.
- **Object Setup** – Lets users select the Shopify and Salesforce objects that will be synchronized, ensuring proper alignment between systems.
- **Field Mapping** – Provides options to map fields between Shopify and Salesforce objects, enabling data consistency during synchronization.
- **Record Types** – Facilitates setting up specific record types for Salesforce objects, ensuring the correct categorization of data.
- **Finish** – Concludes the configuration process, confirming that all steps have been completed successfully, and prepares the system for ongoing synchronization.

Shopify Setup Tab:

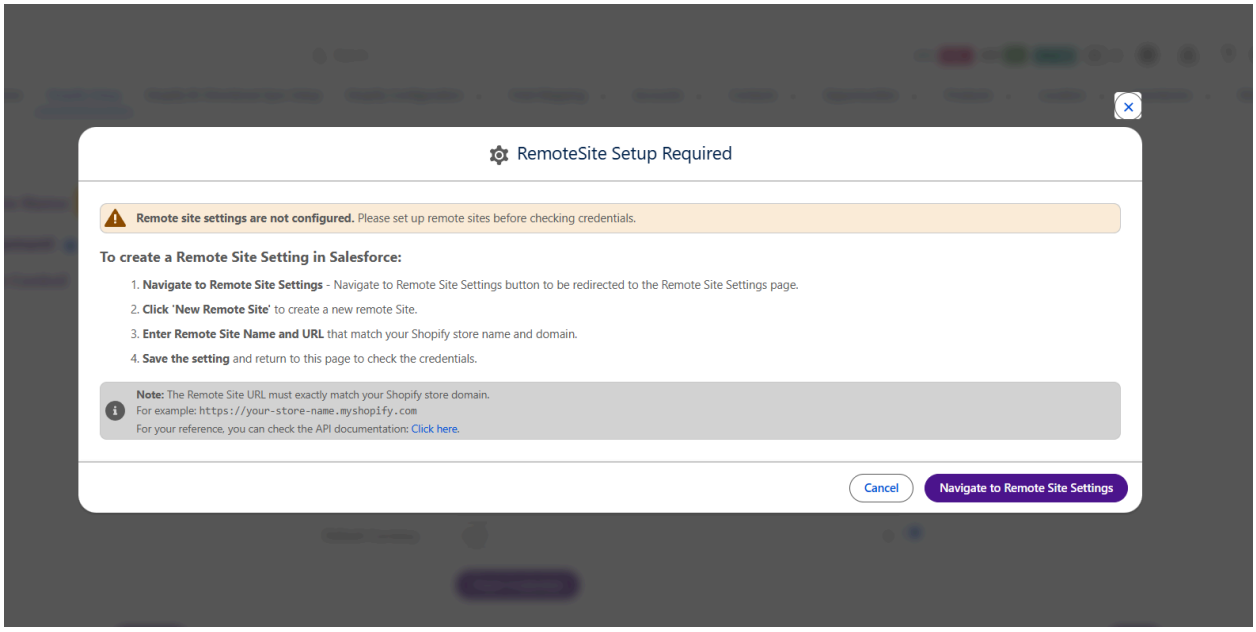
These pages ensure a user-friendly walkthrough for the entire configuration and sync process.

Shopify Credentials Page:



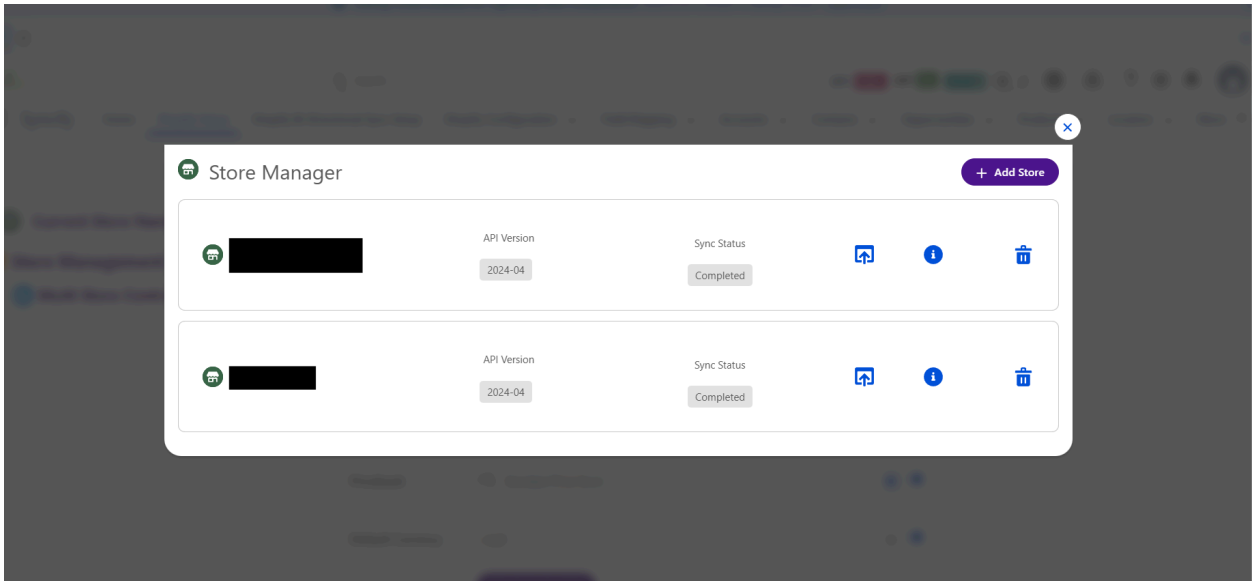
- This section describes the internal logic and backend implementation of the Shopify Credentials Page, where users enter and validate their Shopify store details. These details are then stored in the Shopify Configuration custom object to enable synchronisation with Salesforce.
- When the user clicks “**Check Credentials**”, the store setup is completed, the details are stored in the Salesforce custom object, and the webhooks offered by our application are created in Shopify.
- **Remote Site Settings**:
    - A pop-up dialog box appears, prompting you to create a New Remote Site Setting. The dialog also provides step-by-step instructions to help you complete the setup.
    - After clicking the **"Navigate to Remote Site Settings"** button, it will be redirected to Salesforce's Remote Site Settings page. Provide a **Remote Site Name** and **URL** that matches your Shopify store (e.g., https://your-store-name.myshopify.com).
    - Users will be prompted to create the necessary **Remote Site Settings** in Salesforce to enable secure communication with the Shopify API. We have given the Shopify API reference to provide a **Remote Site Name** and **URL** that matches your Shopify store.

- Save your changes and return to this screen to re-check the credentials. Get back to the **Shopify Credentials** page. Verify the details and click on the Check Credentials button again. The Details should be verified successfully(if the credentials are valid). Now the Shopify Configuration has been done successfully.



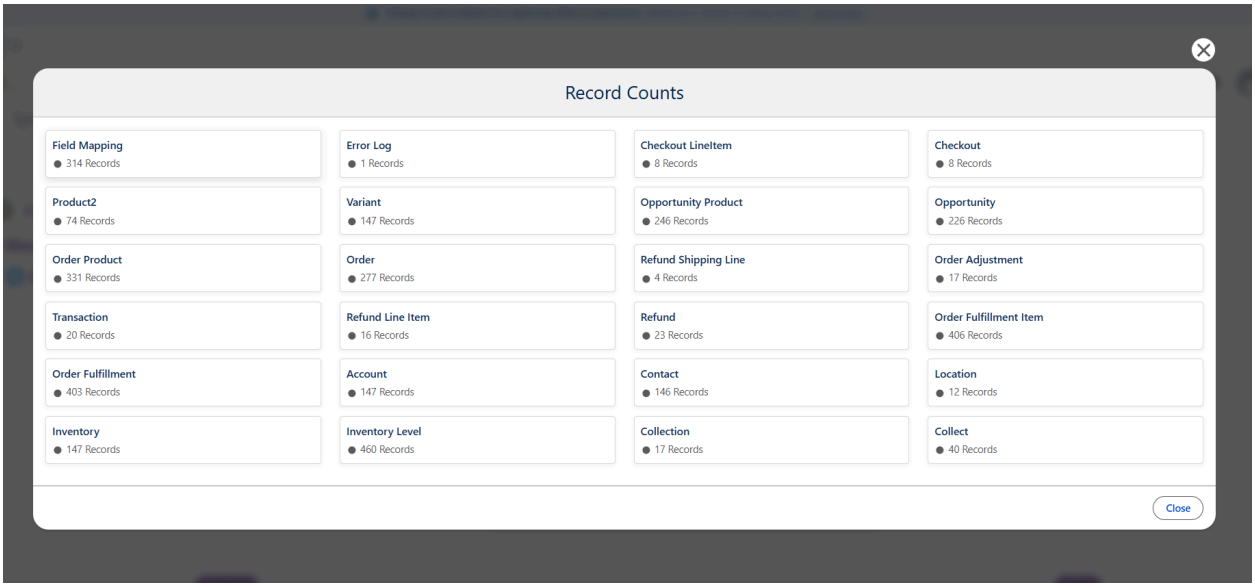
Multi-Store Control:

Once a store is successfully configured through the Shopify Credentials Page, its name appears in the Multi-Store Control section. This section allows users to view, select, and manage all previously configured Shopify stores.



The system dynamically updates this list based on the records created in the Shopify Configuration object, enabling seamless switching and administration of multiple store integrations. When a user clicks the **Multi-Store Control** tab or section, the system performs a backend query (typically via Apex or LWC controller) to retrieve all configured store records from the **Shopify_Configuration__c** custom object. The UI displays the following key elements, populated dynamically from stored metadata:

- **Store Name**
  Retrieved from Shopify_Configuration__c.lcmp__Shopify_Store_URL__c. This identifies each configured store (e.g., *testautoapp*).
- **API Version**
  Fetched from Shopify_Configuration__c.lcmp__API_Version__c. This indicates which version of the Shopify Admin API the store is using (e.g., *2024-04*).
- **Sync Status**
  Derived from Shopify_Configuration__c.lcmp__Is_Batch_Completed__c. This field reflects whether the sync process has started, is in progress, or has completed.
- 📤 **(Open Store Icon)**
  This triggers a navigation event or modal that opens the detailed configuration view for the selected store. It allows editing credentials or triggering sync actions. The store ID or record ID is passed via event or URL parameter.
- ℹ️ **(Imported Records Info Icon)**
  When clicked, this element executes a controller method or SOQL query to retrieve the count of imported records across related objects (e.g., Products, Customers, Orders) and displays the results in a pop-up or modal window.



- 🗑️ **(Delete Store)** – Allows users to delete the configured store, which removes its data and associated Shopify Configuration records from the organization. **Ensure that the Remote Site Setting is deleted manually only after the configured store has been deleted.** Do not delete the Remote Site Setting before deleting the configured store record.

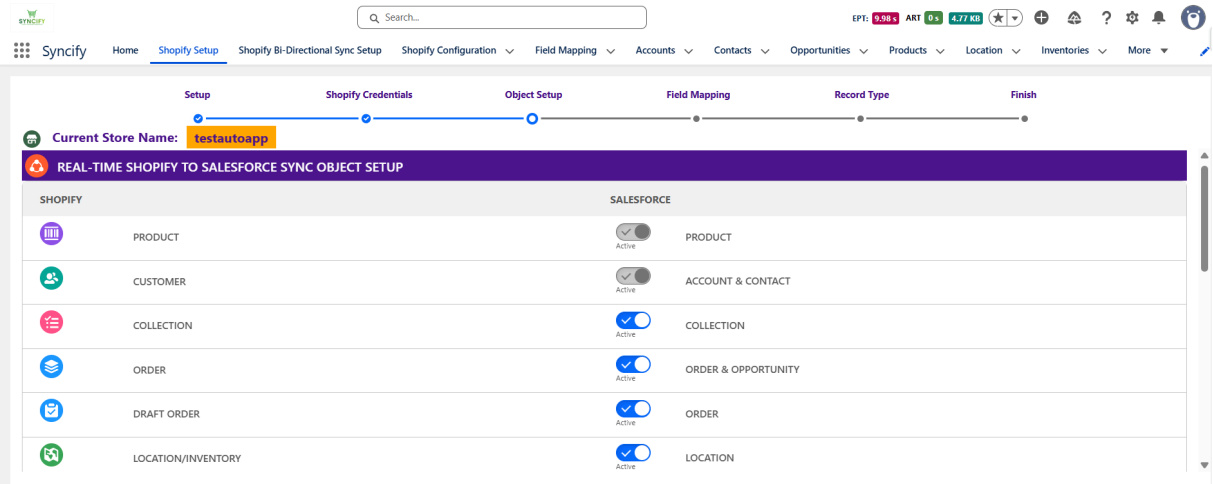
3. Object Setup:

- The **Shopify Object Sync Configuration Page** enables Salesforce administrators to define and control which Shopify object data is synchronised with Salesforce in both **real-time** and during **initial import** operations. This page plays a critical role in establishing object-level synchronisation preferences, ensuring data consistency while allowing fine-grained control over integration behaviour.
- **Left Panel**: Displays available Shopify objects (e.g., Product, Collection, Order).
- **Centre Toggle Button**: Enables or disables synchronisation for each object.
- **Right Panel**: Displays the corresponding Salesforce object(s) that data will sync to.

🧩 Two Sync Configuration Modes

1. REAL-TIME SHOPIFY TO SALESFORCE SYNC OBJECT SETUP

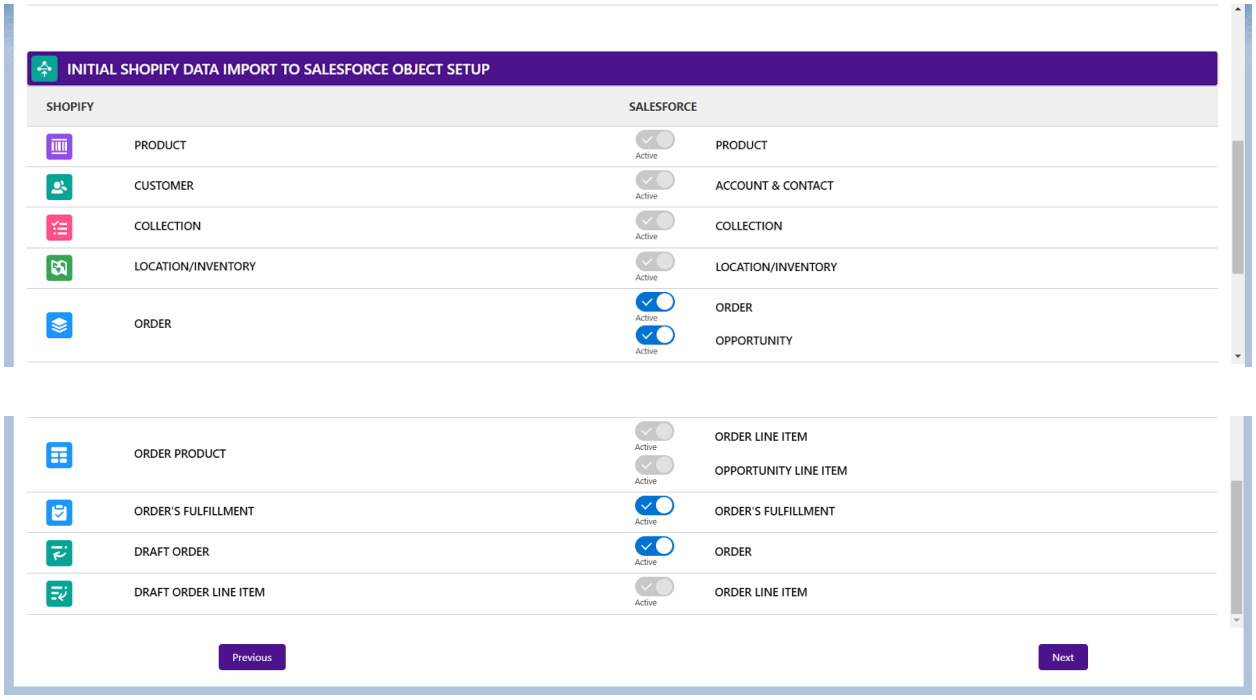
This section governs the real-time data flow triggered by events from Shopify (via webhooks or polling mechanisms). Backend logic determines dependencies and enforces validation rules to ensure minimum requirements are maintained.



✅ System Behavior & Constraints:

- **Non-deactivatable Core Objects**:
  - Product__c, Account, and Contact are always enabled.
  - Logic: UI disables toggles; backend Apex class (ShopifySyncSettingsController) enforces this with validation on save.
- **Configurable Objects**:
  - Collection → Syncs to Collection__c
  - Location/Inventory → Syncs to Location__c, Inventory__c
  - Order → Syncs to Order__c, Opportunity, and Order_Fulfillment__c
  - Draft Order → Syncs to Draft_Order__c

2. INITIAL SHOPIFY DATA IMPORT TO SALESFORCE OBJECT SETUP



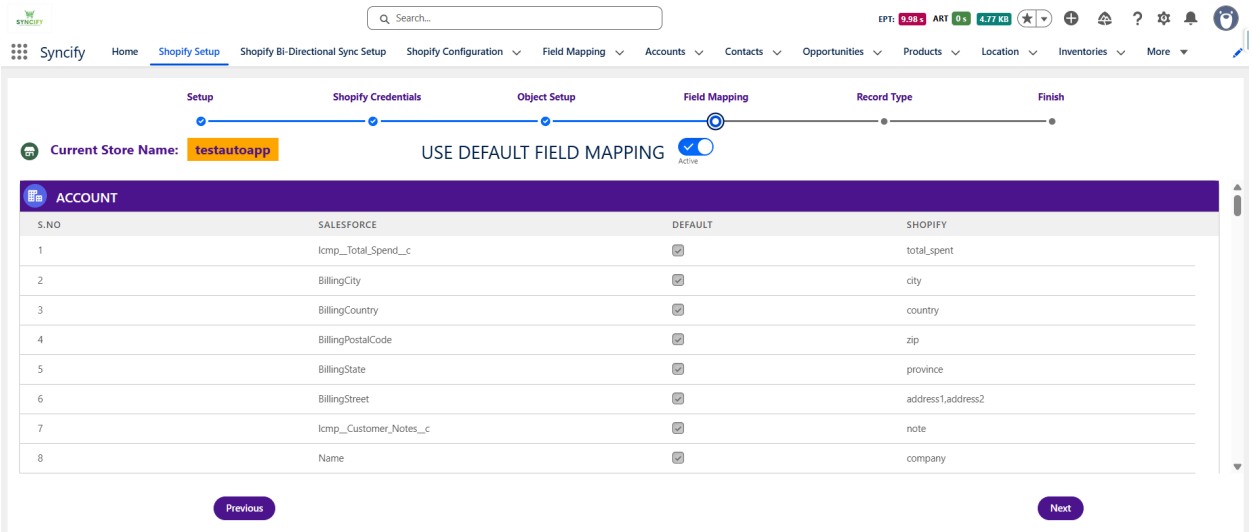
This section defines the object-level data pull configuration used during the **first-time bulk import** from Shopify to Salesforce. It includes both **dependent** and **independent** object toggles, with logic to maintain the integrity of key business records.

✅ Object Dependency Logic:

- **Order Toggle OFF**:
  - Auto-disables Order Line Item, Order Fulfilment
  - Keeps Opportunity and Opportunity Line Item forcibly enabled (disabled UI + Apex enforcement).

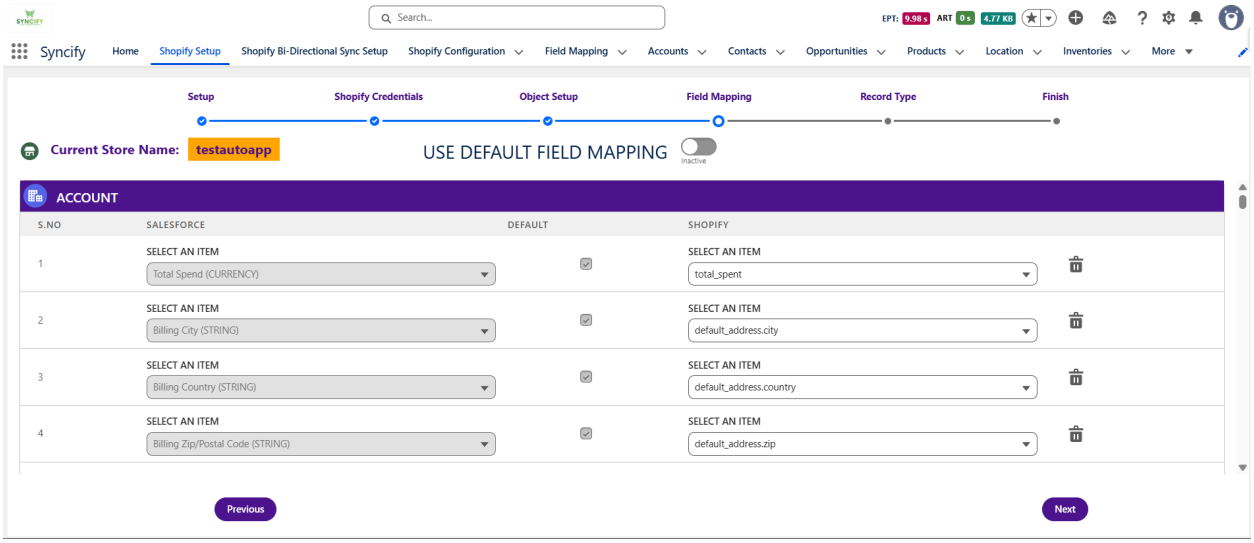
    - Reason: Opportunities serve as fallback containers when orders are disabled.
- **Opportunity Toggle OFF**:
    - Auto-disables Opportunity Line Item
    - Keeps Order and Order Line Item forcibly enabled.
    - Reason: Orders must persist even if Opportunities are not desired.
- **Draft Order Toggle OFF**:
    - Auto-disables Draft Order Line Item.

4. Field Mapping:

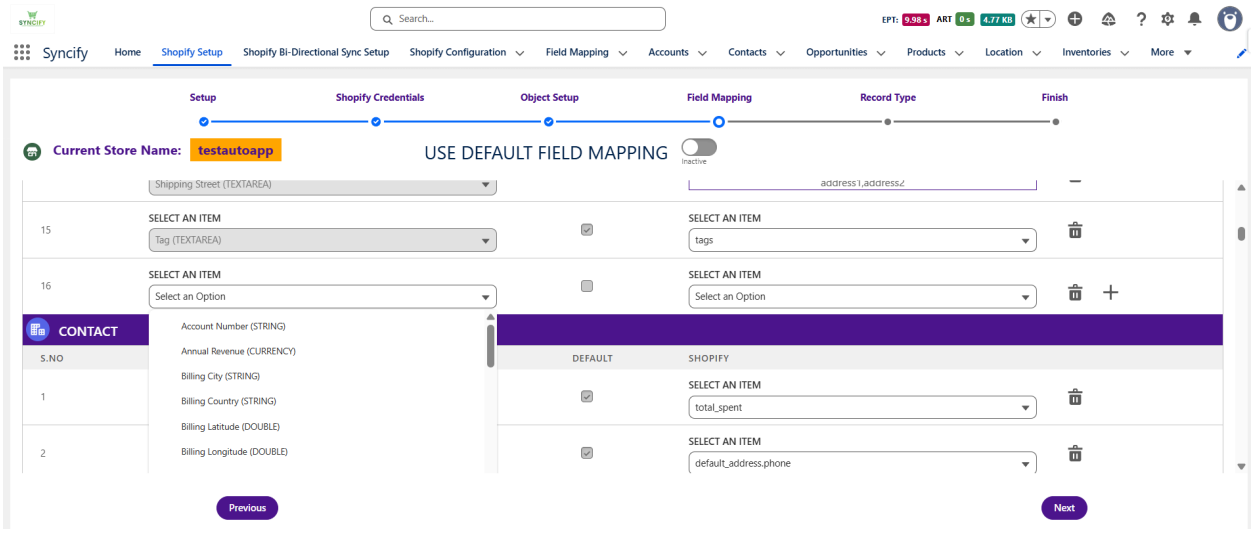


- This page allows users to configure the fields for specific objects that were previously configured in the Object Setup page. The **Field Mapping** page includes the following elements:

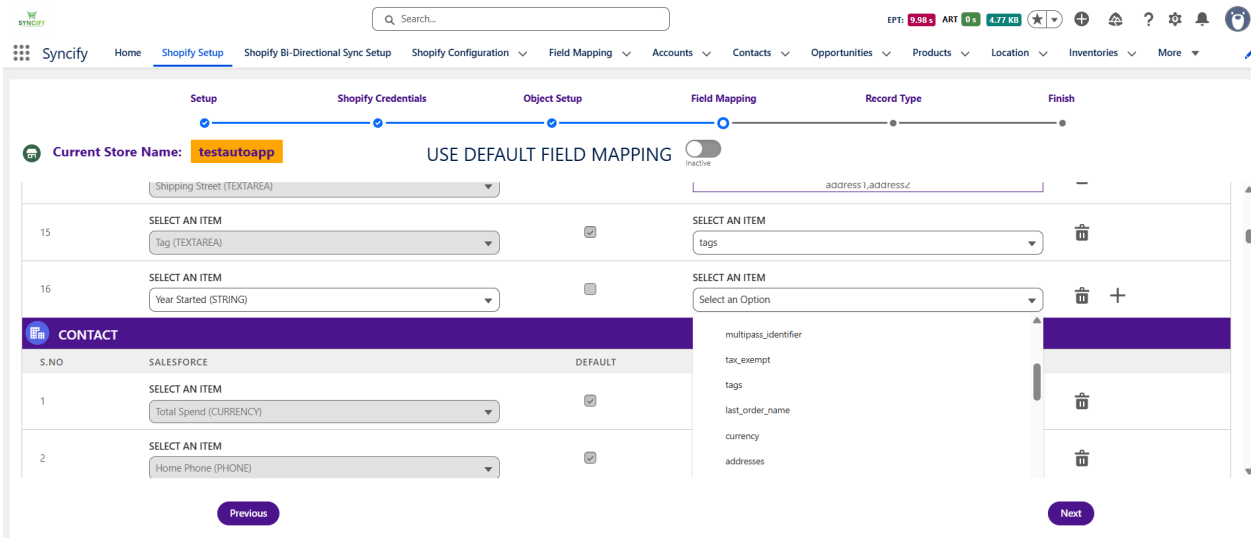
    - **USE DEFAULT FIELD MAPPING toggle**: By default, this toggle is active, enabling predefined fields for the objects. If users wish to customise the field mapping, they can do so by turning this toggle off to enable manual configuration.
    - **Object Name**: Displayed in the left corner, showing the name of the object currently being configured.
    - **Salesforce Field Section**: Located on the left side, displaying the Salesforce fields that will be mapped to Shopify.
    - **Shopify Field Section**: Located on the right side, displaying the Shopify fields that correspond to the Salesforce fields.



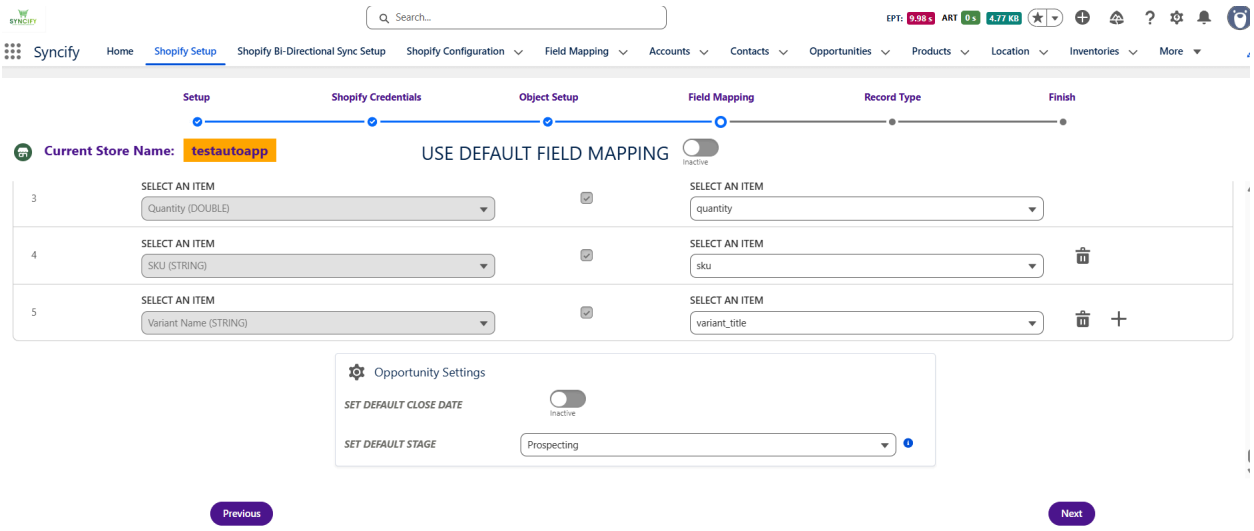
- This page provides flexibility for users to either use the predefined field mappings or customise the field mappings as needed for each object.
- Users have the option to either use the default field mapping or customize the field mapping by deactivating the **USE DEFAULT FIELD MAPPING** toggle. Once the toggle is inactive, the following actions are available:
    - **ADD New Fields**: Users can add new fields to the field mapping configuration.
    - **EDIT Existing Fields**: Users can modify the existing field mappings.
    - **DELETE Pre-defined Fields**: Users can remove any of the pre-defined fields from the mapping.
- This provides users with full control to customise field mappings based on their specific data integration needs.
- There exist the following elements exist:
    - ➕**(Add new Field Mapping Row):** To add a new Salesforce to Shopify Field Mapping Row.
    - 🗑️**(Delete Field Mapping Row):** To delete the present or existing Salesforce to Shopify Field Mapping Row.
    - **Left - Dropdown Box:** Displays only the Salesforce fields of that object.
    - **Right - Dropdown Box:** Displays the related Shopify fields of that object.
- The **Salesforce** section displays the fields along with their data types for the selected object. When a field is chosen from this section.



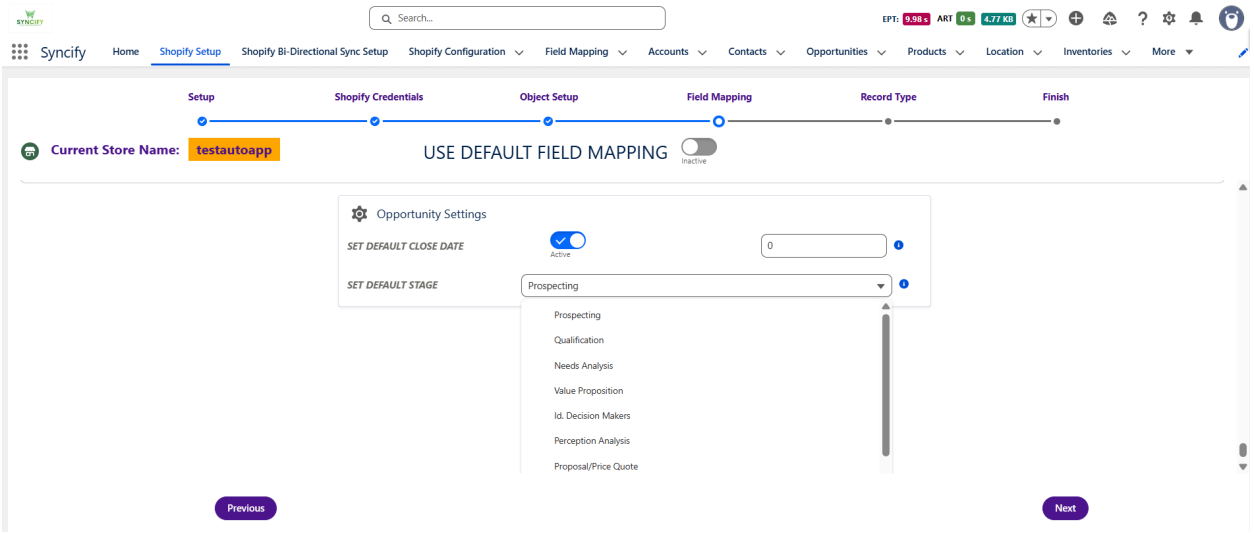
- The **Shopify** section only shows fields that can match the selected data type for proper field mapping.
- However, if a text or text-related field is selected from the Salesforce section, almost all Shopify fields will appear for mapping since text fields are more universally compatible.
- In rare cases, no fields will appear in the Shopify section if an incompatible data type is selected from Salesforce, meaning there are no matching Shopify fields available for mapping.
- Duplicate field mappings are not supported on the Salesforce side, meaning a single Salesforce field cannot be mapped to multiple Shopify fields. However, the reverse is allowed—users can map the same Shopify field to multiple Salesforce fields.



- This flexibility on the Shopify side enables broader mapping options while ensuring data integrity on the Salesforce side by preventing duplicate mappings and avoiding data type conflicts.

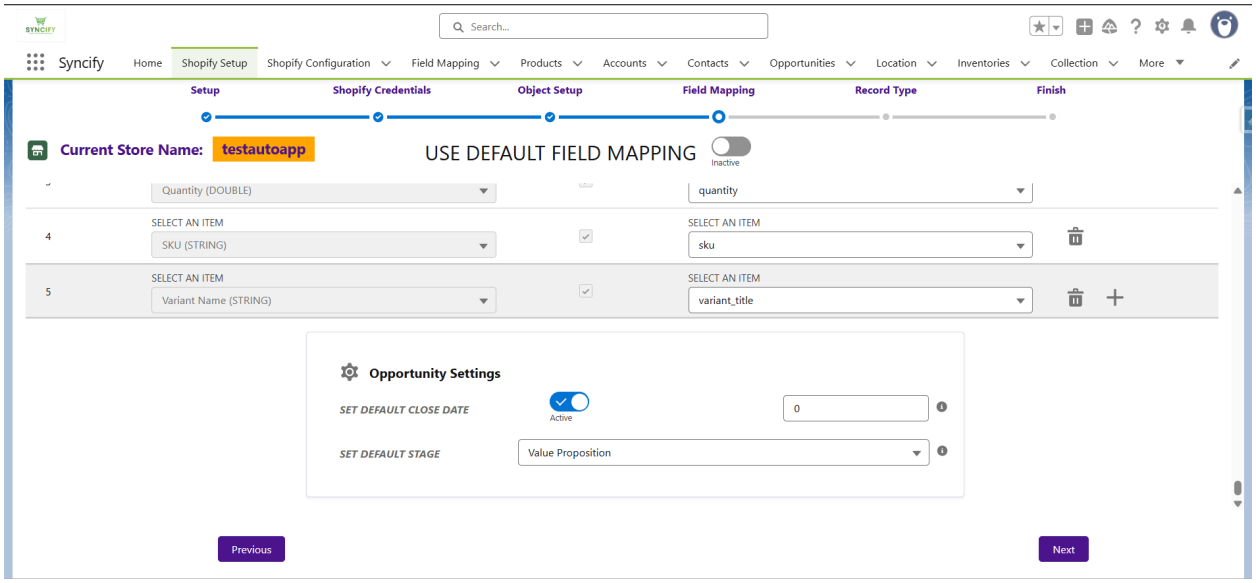


- **Configure the Opportunity Settings**:
  - If you are using Custom Field Mapping by deactivating the USE DEFAULT FIELD MAPPING toggle, you have the option to configure the ⚙️ Opportunity Settings (optional). Follow these steps:
  - Scroll to the Bottom of the Field Mapping Page: Locate the ⚙️ Opportunity Settings section.



  - **Toggle On "GIVE DEFAULT CLOSE DATE"**: Enable this option to assign a default close date to the Opportunity records.

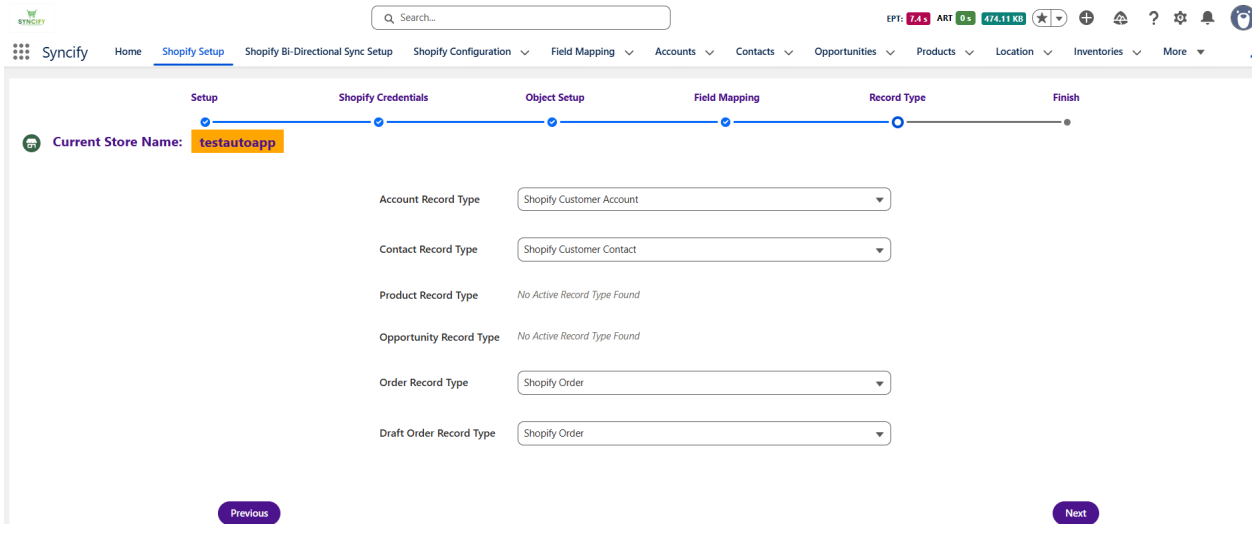
- **Specify the Number of Days**: Enter the number of days to be added to the close date for the Opportunity record.



- **Specify the DEFAULT STAGE**: Set the default value for the **StageName** field, which will appear in the Opportunity record.

- These are the features and add-on settings that can be done in the Field Mapping Page.

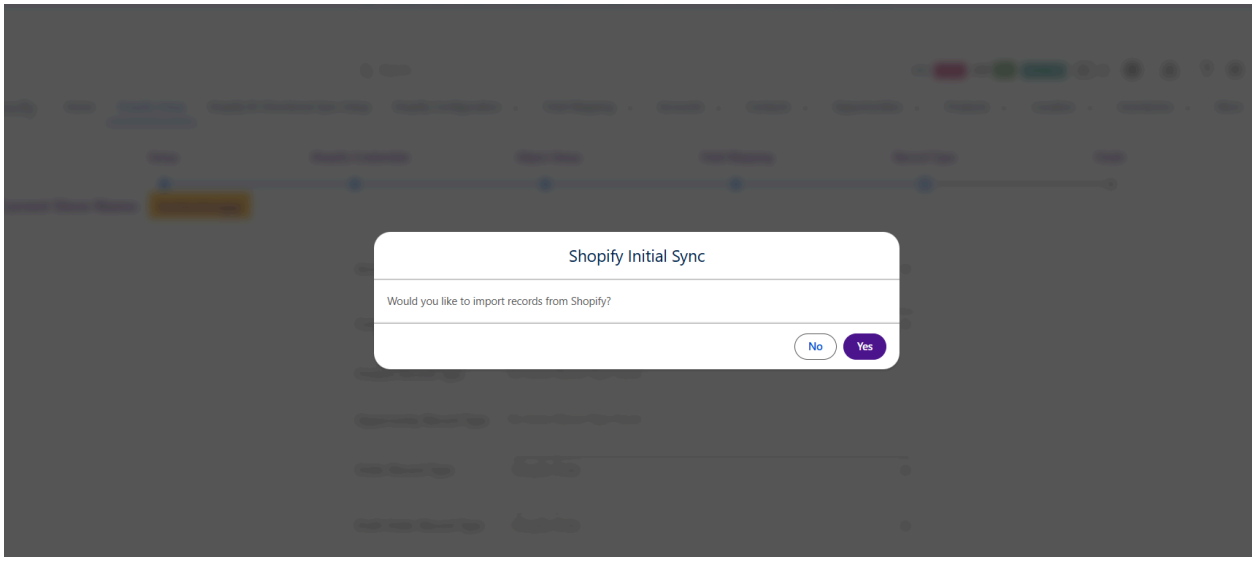
5. Record Types:



- This page ensures the master objects' **Record Type**, which segregates the records being imported from Shopify into Salesforce. The following elements are present on the **Record Type** page:

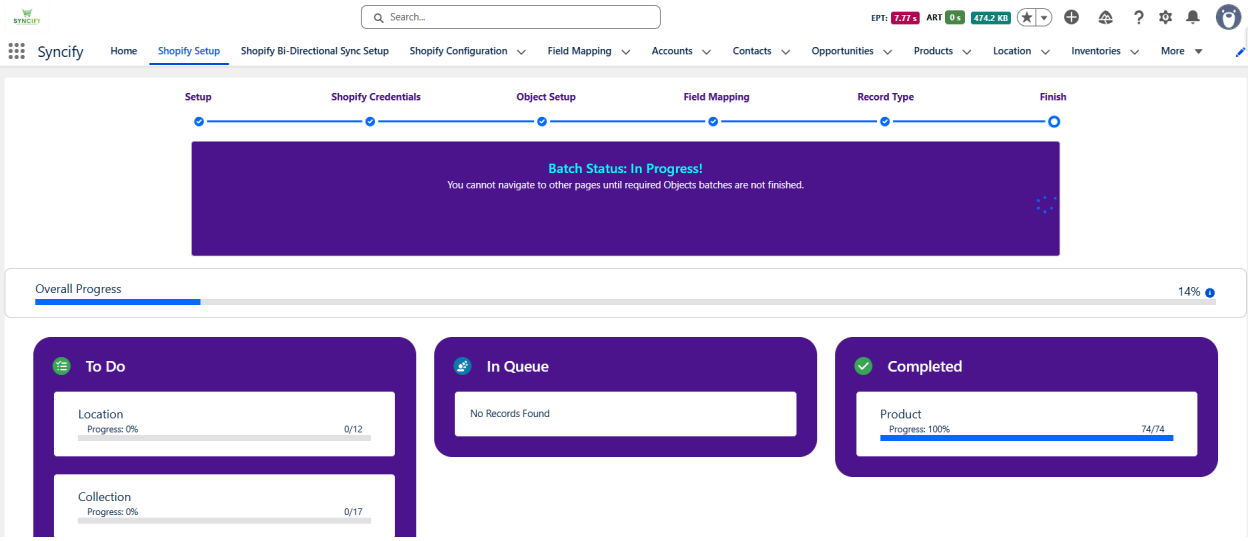
    - **<Object_Name> Record Type**: Displays the name of the object for which the record type is being configured.
    - **Drop-Down Box**: Contains a list of available record types for the selected object.
- By selecting the appropriate record type, users can properly organise and categorize the imported records within Salesforce, ensuring a more structured data import process.

6. Finish:

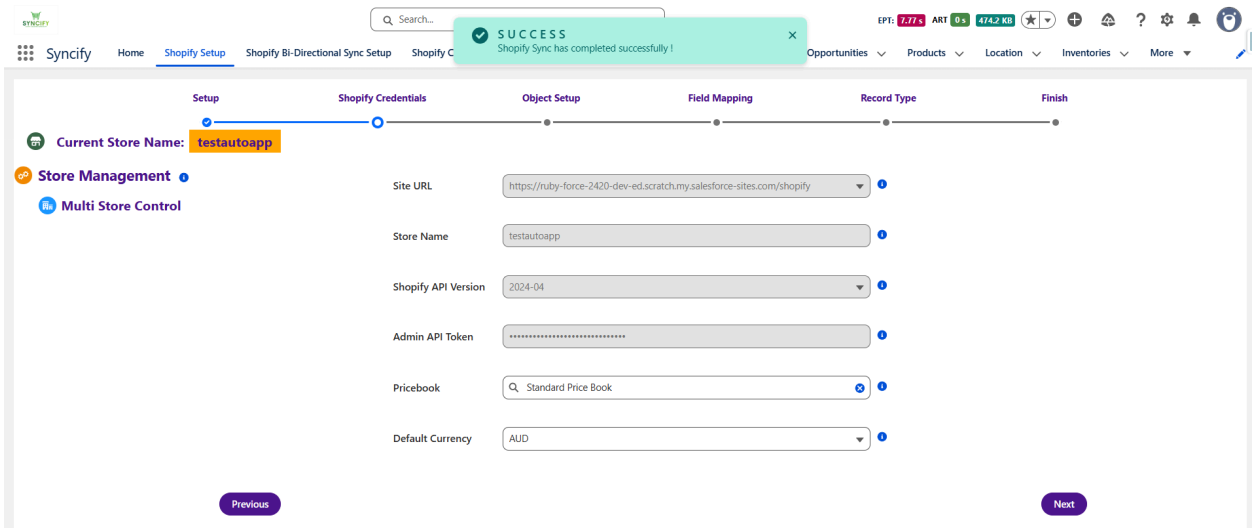


- Before navigating to the final page, a small confirmation pop-up appears, asking whether you would like to proceed with importing the records from Shopify.
- Upon clicking the **Yes** button, the system immediately begins importing the records, and you are redirected to the final page.
- The **Finish** page contains three layers for monitoring the data import process:
    - **To-Do**: Lists the objects queued for data import.
    - **In Queue**: Shows the current progress of objects being imported.
    - **Completed**: Displays the objects for which the data import process has been completed.
- Apart from the layers mentioned, additional necessary components are included for enhanced tracking of the import process:
    - **Overall Progress**: Displays the overall progress of the entire object synchronisation process, helping users track the total sync status.
    - **Specific Object Progress**: Shows the progress for each specific object being synced, allowing users to monitor the sync status of individual objects.

- While the records are being imported, the users are not allowed to navigate to any of the pages.

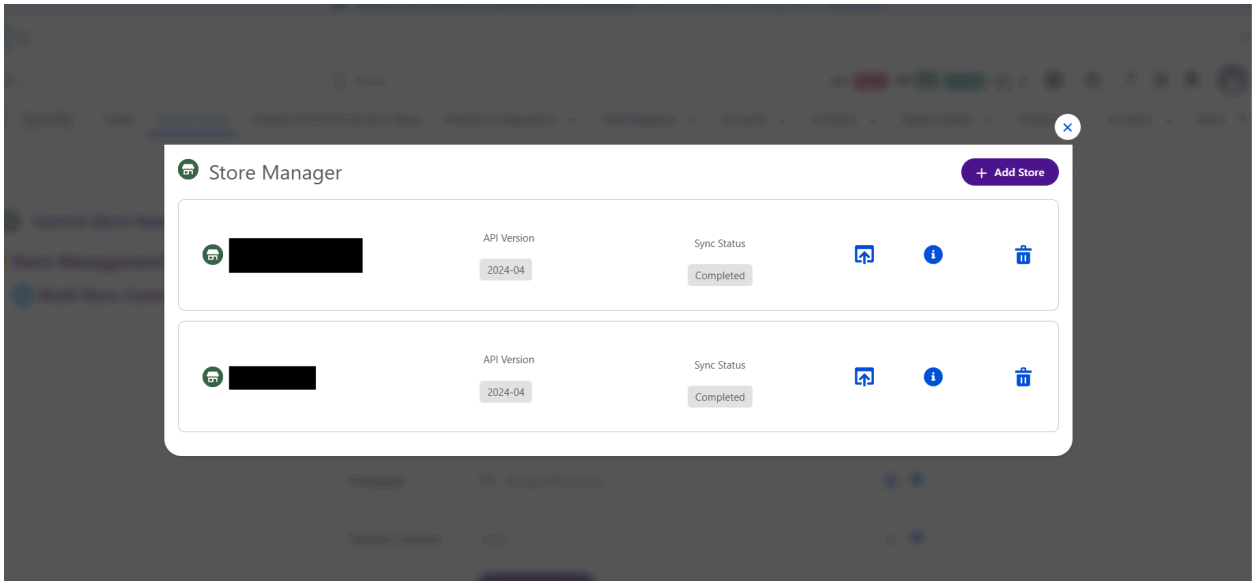


- Once the **Overall Progress** reaches 100%, the system remains on the **Finish** page because some of the child and dependent object records are still being imported. The batch process will only be considered complete when all dependent object records have been successfully imported.
- After the entire batch is completed, including the dependent records, the system automatically navigates back to the **Shopify Credentials** page, signalling the successful completion of the import process. This ensures that all related data is fully synchronised between Shopify and Salesforce.

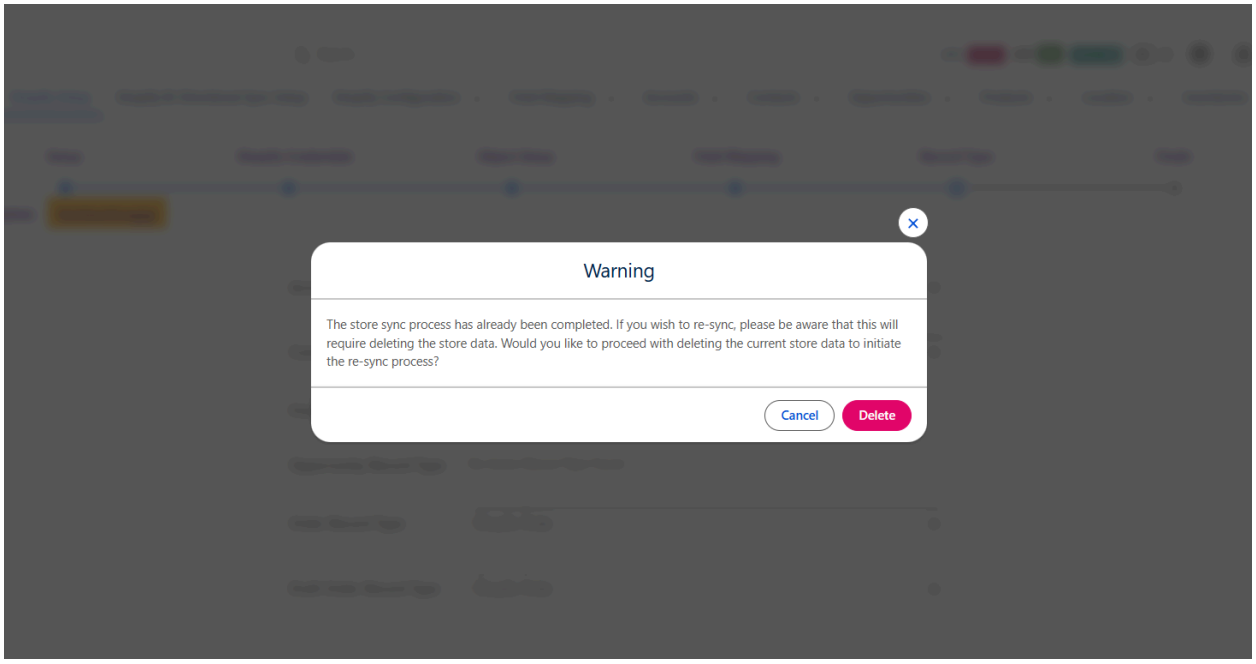


- Users can verify whether the batch process has completed by navigating to the **Shopify Credentials** page and clicking on the **Multi-Store Control**. Under the

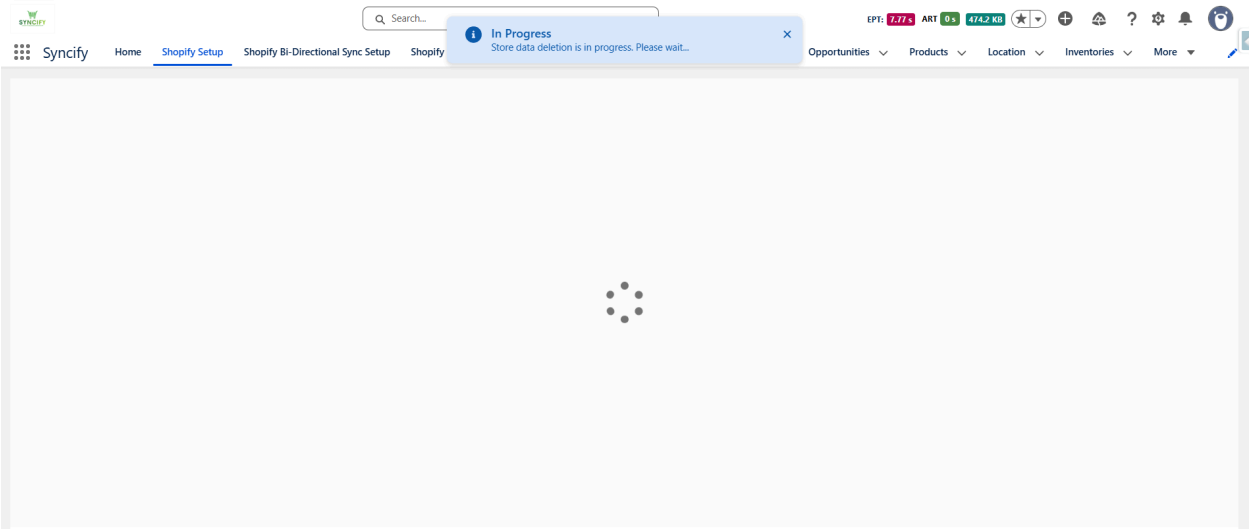
Store Information section, users can check the sync status of the store to confirm if the batch import and all dependent records have been successfully synchronised between Shopify and Salesforce.



- This provides a quick and convenient way for users to monitor the completion of the Shopify data sync.
- When a user attempts to sync an already synced store, a pop-up window will appear, prompting them to delete the existing data.

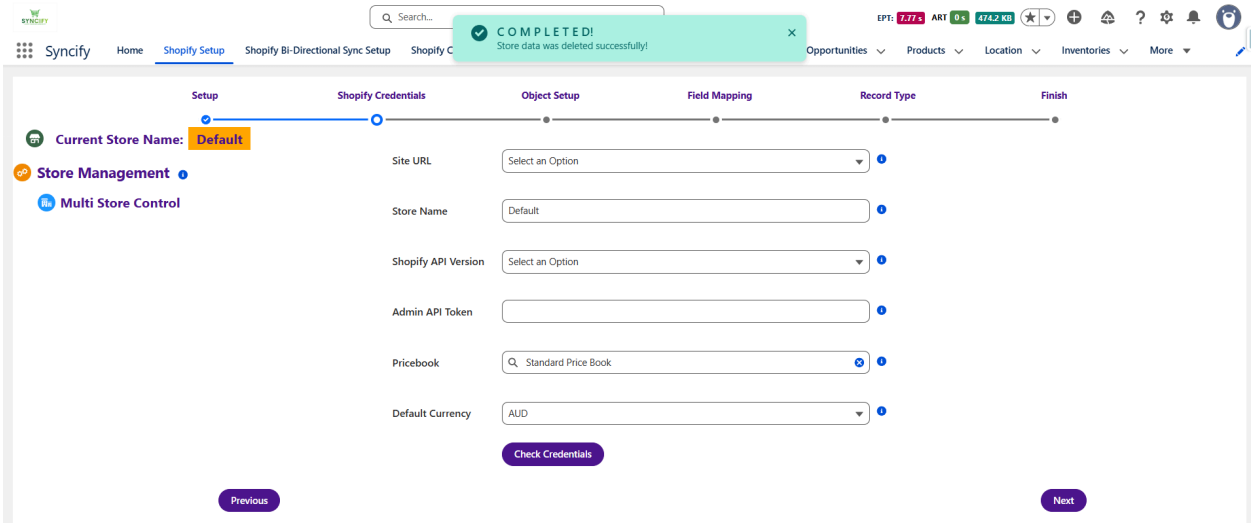


- Upon confirming the deletion of the current records, the deletion progress will run once it is completed, and users will be able to re-sync the store by configuring it from the beginning. Allows users to delete the configured store, which removes its data and associated Shopify Configuration records from the organisation. **Ensure that the Remote Site Setting is deleted manually only after the configured store has been deleted.** Do not delete the Remote Site Setting before deleting the configured store record.



These are the components and available elements in the **Syncify App**, which ensure seamless data synchronisation.

- They enable the smooth import of records from the Shopify store into the Salesforce Org, allowing users to manage data effortlessly and maintain real-time updates across both platforms.



One-Time Data Migration:

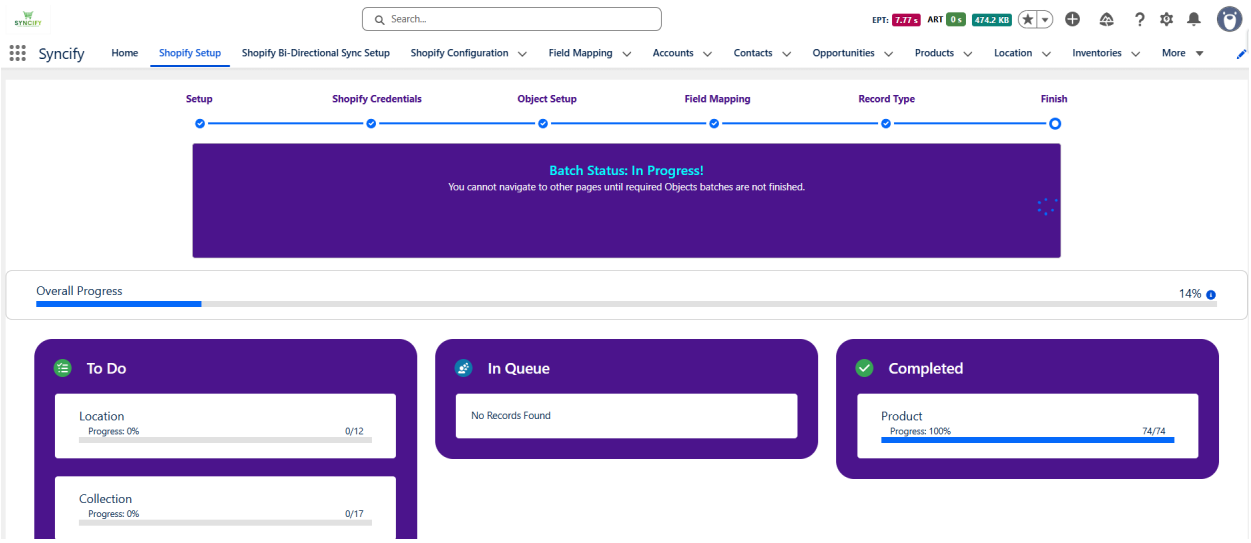
The **One-Time Data Migration** process is designed to import all required historical records from Shopify into Salesforce in a single operation. This feature is typically used during the initial setup or for migrating a large set of data at once. The following steps and components are involved in the one-time migration:

1. **Data Selection**:
   Before initiating the migration, users will have the option to select which data (such as products, orders, customers, etc.) to import from Shopify into Salesforce.
2. **Confirmation Pop-Up**:
   A confirmation pop-up will appear, asking the user to confirm if they want to proceed with the one-time migration. This ensures that users are aware that the process may take some time, depending on the volume of data being migrated.
3. **Progress Monitoring**:
   The **Finish** page will display the same progress layers (To-Do, In Queue, Completed), showing the status of the one-time migration. Users can track the overall progress and specific object progress to monitor the sync status.

   A summary of the migration process will be available after completion, helping users quickly verify the successful import of the data.
4. **Post-Migration Actions**:
   - Once the migration is completed, users will be automatically redirected to the **Shopify Credentials** page, where they can check the sync status of the migrated store.
   - If any issues arise during migration, users will be notified and given the option to resolve them.
5. **Data Integrity**:
   The system ensures data integrity by importing only the necessary and valid records, excluding duplicates and incompatible data types. Once the migration is completed, all records will be synchronised and available for further processing in Salesforce.

Reload Operation:

During the **Initial Sync** process, a **batch sync** is initiated to import data from Shopify into Salesforce. The system ensures that data synchronisation happens seamlessly, but it also restricts navigation during the batch job to ensure that no conflicting actions are taken while the sync is in progress.



Key Features of the Reload Operation:

1. **Page Refresh or Navigation Attempt**:
   While the Initial Sync is in progress, if the user attempts to refresh the page or navigate to another page within the Shopify Setup tab, the system will not allow the transition. Instead of blocking the action, the page will automatically reload and redirect the user back to the sync page to continue displaying the current sync progress.
2. **Sync Enforcement**:
   This behaviour ensures that the user remains on the sync page throughout the batch process, preventing disruption to the data import and ensuring all object and dependent object records are correctly synchronised.
3. **Batch Job Execution**:
   The system runs an Apex batch job for the sync process. Once this batch job begins, it ensures all records are being processed in a systematic way. During this time, the user will see a progress indicator for the overall sync status, and individual object progress will also be displayed.
4. **Post-Sync Navigation**:
   After the batch sync is completed and all records are successfully imported, the page navigation restrictions will be lifted. At this point, the user can proceed to any page within the **Shopify Setup** tab or navigate to other sections of the application.

Error Handling During Initial Sync

During the **Initial Sync** process, data is imported from Shopify into Salesforce using **Apex batch jobs**. This process heavily relies on **accurate field mappings** between Shopify and Salesforce. Any discrepancy in these mappings may result in insertion failures, which are systematically tracked.

Error Logging

All errors encountered during Initial Sync are captured and stored in a custom object named **ErrorLogs**. Each error log record typically includes:

- **Object Name**: The Salesforce object where the error occurred
- **Error Message**: Description of the error (e.g., "Required field missing", "Picklist value not found")
- **Stack trace:** Optional: Stack trace for debugging.
- **Store Name**: Identifies which Shopify store the error came from (important for multi-store environments)
- **Class Name & Method Name:** Apex class where the error occurred and the Specific method in the Apex class.

These logs give users full visibility into issues and assist in resolving data mismatches prior to reinitiating the sync.

Common Causes of Errors

1. **Missing Field Mapping**
   Required Salesforce fields are not mapped to any Shopify fields.
2. **Incorrect Data Types**
   Data type mismatches between Shopify and Salesforce fields (e.g., Text to Number).
3. **Validation Rule Failures**
   Salesforce validation rules trigger due to incomplete or invalid data.
4. **Lookup Field Failures**
   Referenced records (e.g., Parent object) do not yet exist in Salesforce.
5. **Duplicate Rule Violations**
   Duplicate management settings block insertion of new records.
6. **Required Data Missing in Shopify**
   Shopify payload lacks data for Salesforce-required fields.
7. **Invalid Picklist Values**
   The payload includes values not listed or activated in Salesforce picklists.

Handling Unsupported Field Types in Field Mapping

(For example: Sendoship App Integration) - Certain Salesforce field types are not directly supported by the Shopify API payload and thus may cause issues when mapped via the Syncify App field mapping screen. These include:

- Currency Fields
- Picklist (Single or Multi-Select)

Recommended Workarounds

1. **Use Apex Triggers or Salesforce Flows**
   Automate logic **after sync completion** to handle:
   - Conversion of string values into valid picklist entries
   - Currency calculations or transformations
2. **Make Fields Optional**
   If the field is marked as required in Salesforce, consider:
   - Removing the required flag
   - Assigning default values through automation
3. **Create Compatible Proxy Fields**
   - Add new **Text or Number** fields for mapping in Syncify
   - Use Flows or Triggers to populate the actual Picklist or Currency fields afterwards

> ⚠️ **Note:**
> This limitation applies to **all custom apps**, not just Sendoship. Any field type unsupported by Shopify's data model must be handled using one of the strategies above.

Summary

- Errors during Initial Sync are logged in Error Logs for easy diagnosis.
- Proper field mapping and data format compatibility are essential.
- Unsupported fields should be managed through post-sync automation or proxy fields.
- Multi-store environments benefit from identifying errors by store via logs.

This ensures that data from Shopify is imported cleanly and Salesforce records remain accurate and complete.

Real-Time Synchronisation:

1. Overview:

- Once the Salesforce org is configured and the connection with Shopify is established, webhooks are automatically created before the data import process.
- Webhooks remain live and functional unless the store configuration is deleted completely from the Salesforce org.
- Webhooks are responsible for real-time synchronisation and trigger the following events:
    - **object_record/create**: When a new record is created.
    - **object_record/update**: When an existing record is updated.
    - **object_record/delete**: When a record is deleted.
    - **object_record/connect**: When a record is connected.
    - **object_record/disconnect**: When a record is disconnected.
    - **object_record/read**: When a record is read.
- These events ensure seamless real-time data sync between Shopify and Salesforce, keeping the data consistent.

2. Available Objects:

SNo	Shopify Object	Salesforce Object
1	Product	Product2
2	Variant	lcmp__Variant__c
3	Collection	lcmp__Collection__c
4	Collect	lcmp__Collect__c
5	Location	lcmp__Location__c
6	Inventory	lcmp__Inventory__c
7	Inventory Levels	lcmp__Inventory_Level__c
8	Customer	Contact
9	Customer	Account
10	Order	Order

11	Order Item	OrderLineItem
12	Opportunity	Opportunity
13	Order Item	OpportunityLineItem
14	Checkout	lcmp__Checkout__c
15	Checkout LineItem	lcmp__Checkout_LineItem__c
16	Order Fulfillment	lcmp__Order_Fulfillment__c
17	Fulfillment Item	lcmp__Order_Fulfillment_Item__c
18	Refund	lcmp__Refund__c
19	Refund Line Item	lcmp__Refund_Line_Item__c
20	Refund Shipping Line	lcmp__Refund_Shipping_Line__c
21	Order Adjustment	lcmp__Order_Adjustment__c
22	Transactions	lcmp__Transaction__c

2. Field Mapping Setup:

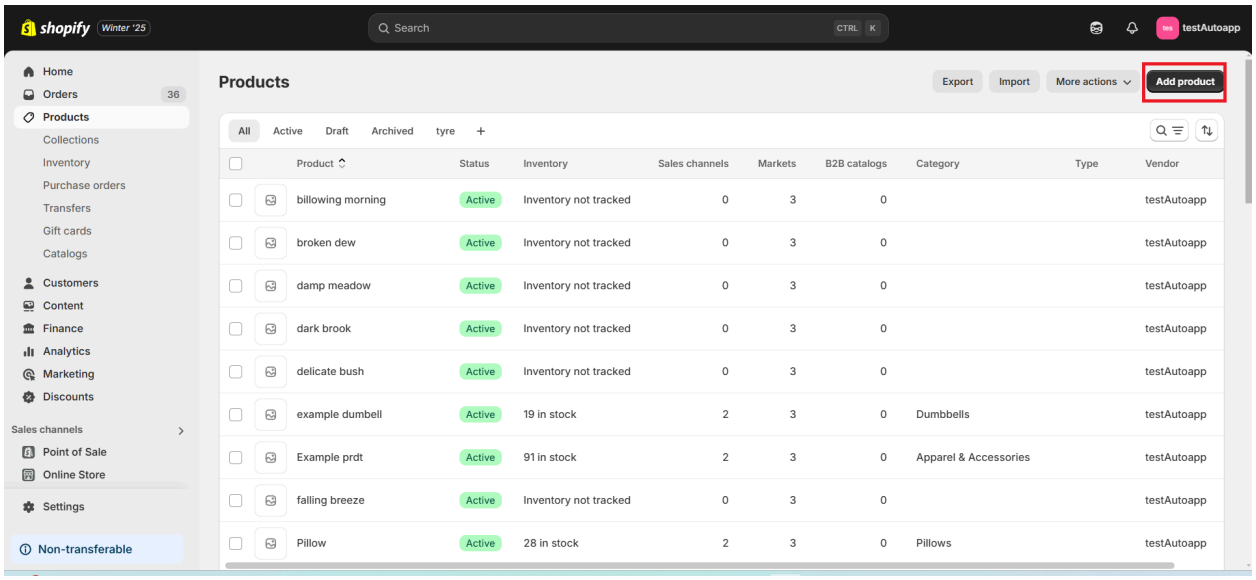
- These are the available fields, which are mostly in use for the above-defined objects: Shopify to Salesforce Field Mapping

3 . Example:

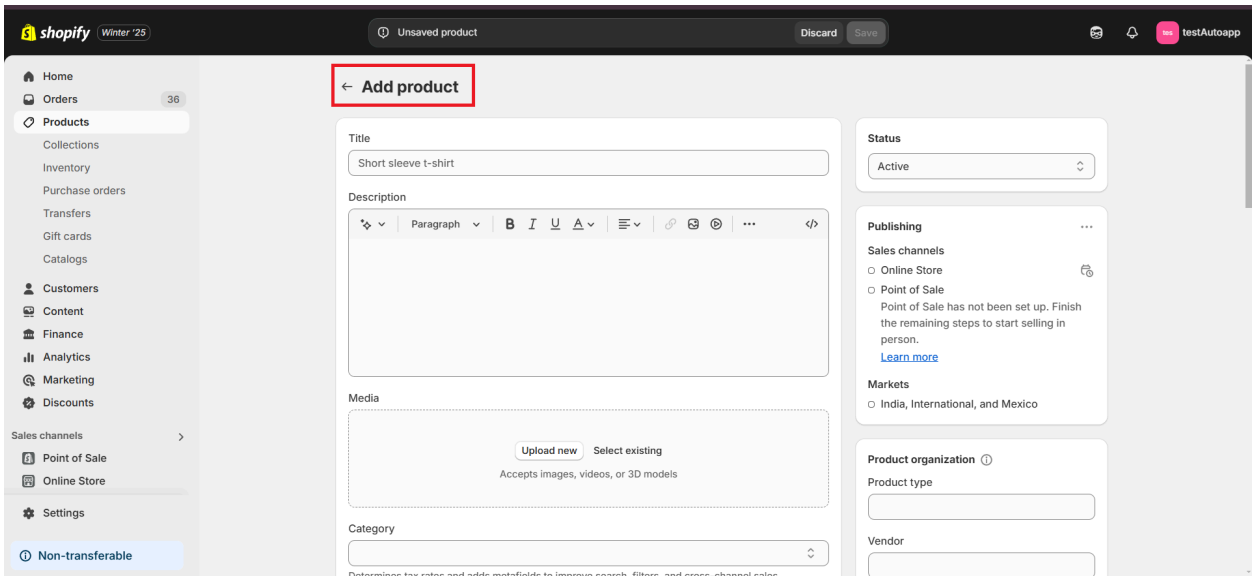
Creation of Records:

1. **Create a Record in Shopify:**
    - Go to any of the supported objects (e.g., Products, Customers, Orders) in your Shopify store.
    - Click on the **Add** button to create a new record.
    - Fill in the required details for the record.
    - Click on **Save** to create the record in Shopify.
2. **Sync to Salesforce:**
    - The same record, with matching details, will be automatically created in Salesforce through the real-time sync process.

3. **Verification:**
    - To verify, navigate to the corresponding object's record page in Salesforce (e.g., Products, Accounts, Opportunities).
    - Ensure the record is correctly created with the same details as in Shopify.
4. **Example:**
    - The following example provides a demonstration of Product creation.
        i. Navigate to the Product section in the Shopify Store
        ii. Click on the **Add Product** button.

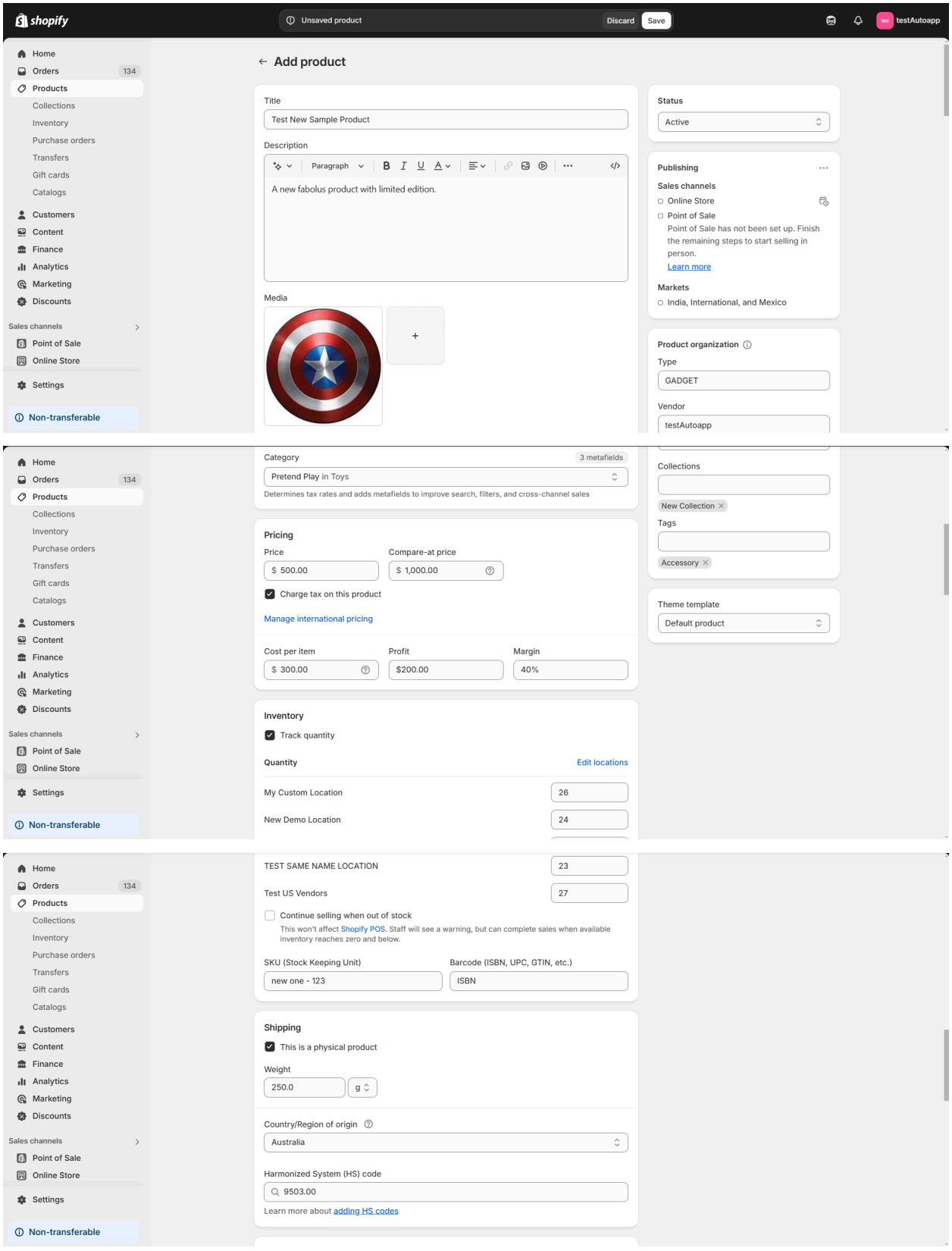


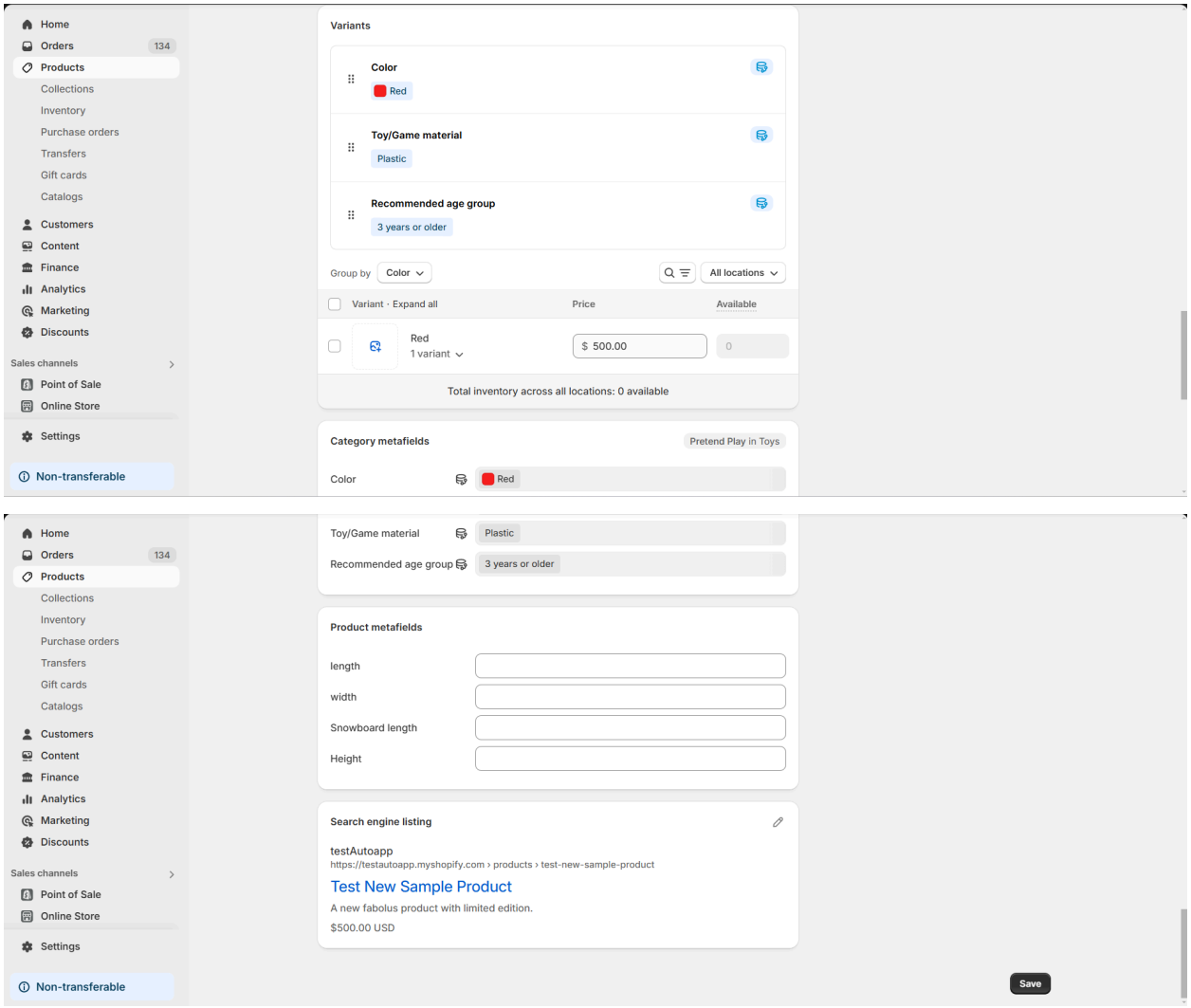
iii. Another page section opens for creating a new product.



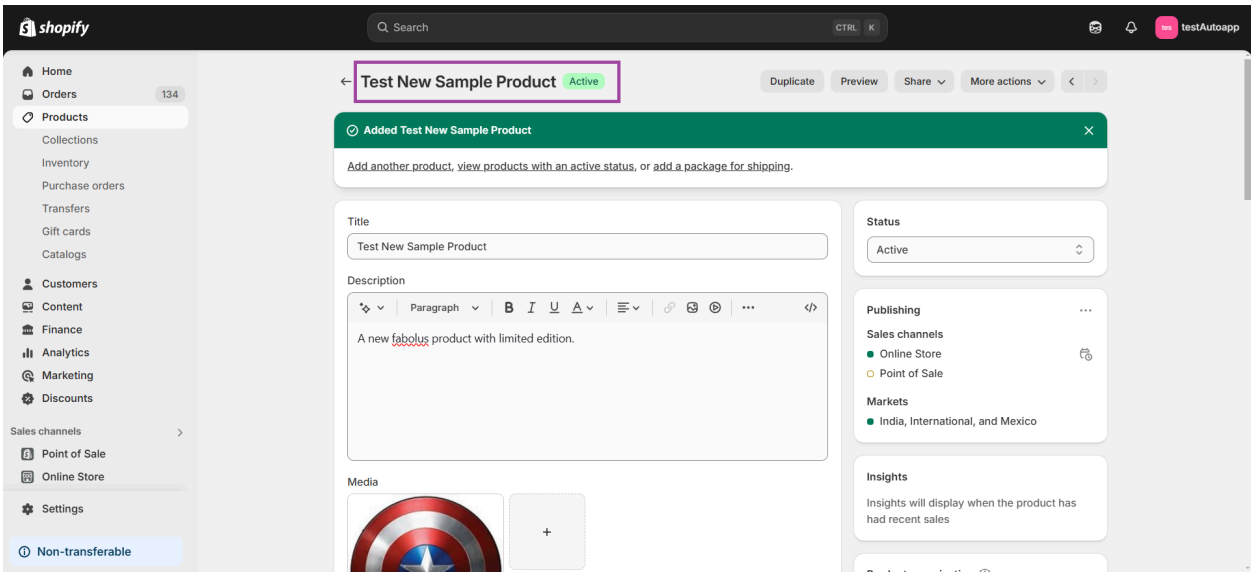
iv. Fill in the necessary fields

v. Click on the **Save** button above the page

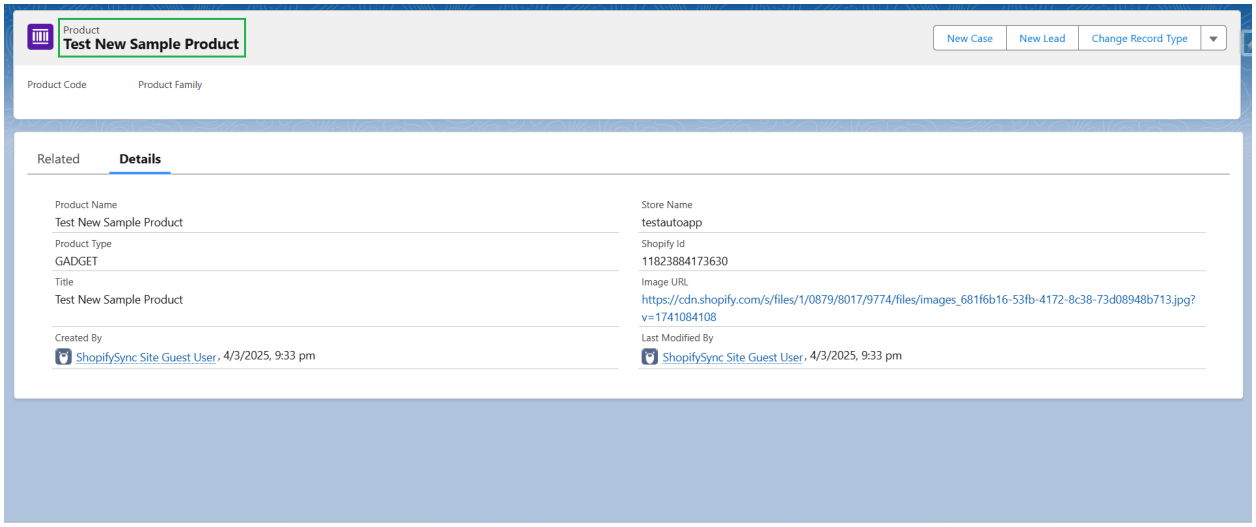




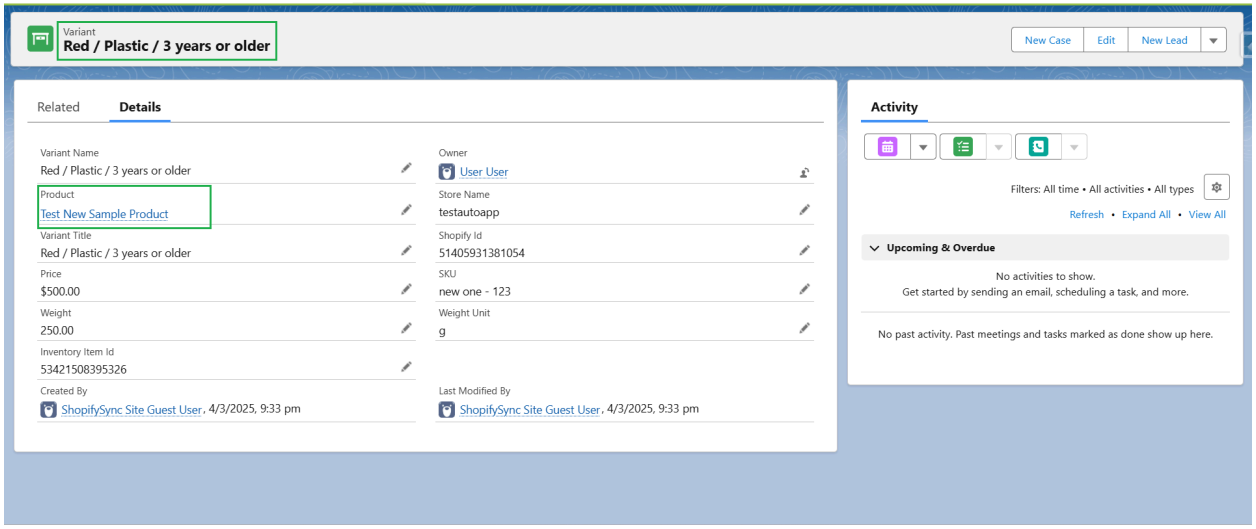
vi. A new Product has been added to the Product List.



- A new Product with the same details will be created in the Salesforce org.



- A relative variant is also created with this product.



- Verify the creation of the new Product record on the **Product2** record page.
- The same creation process applies to the other supported objects for synchronisation.

Update of Records:

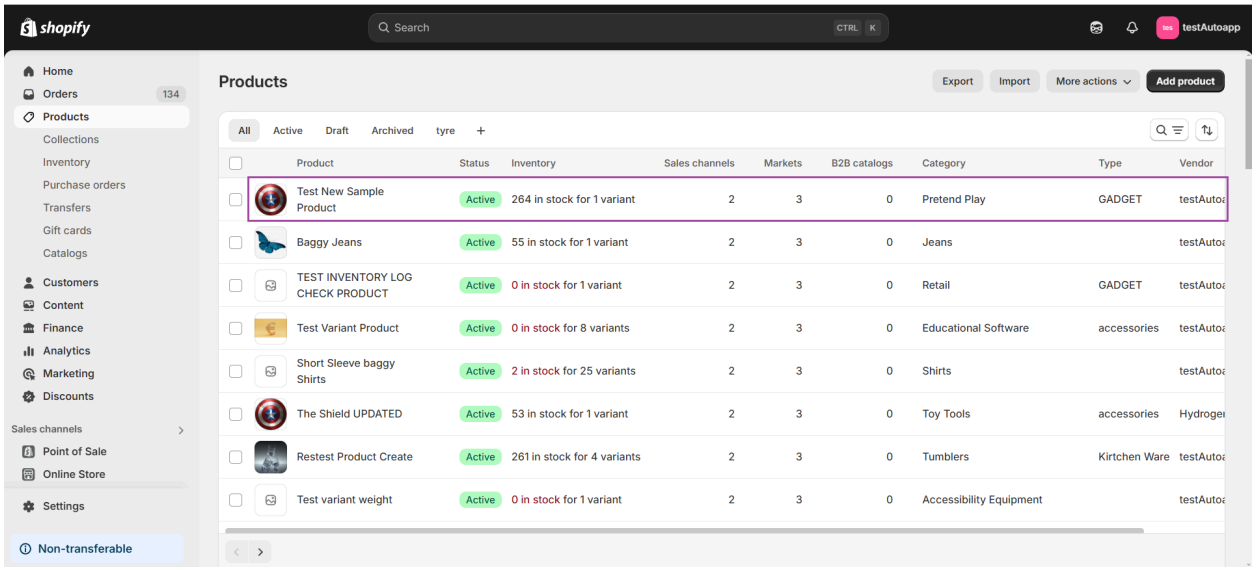
1. **Update a Record in Shopify:**
   - Navigate to any of the supported objects (e.g., Products, Customers, Orders) in your Shopify store.

- Select the record you want to update.
- Modify the necessary fields as needed.
- Click on **Save** to apply the updates.

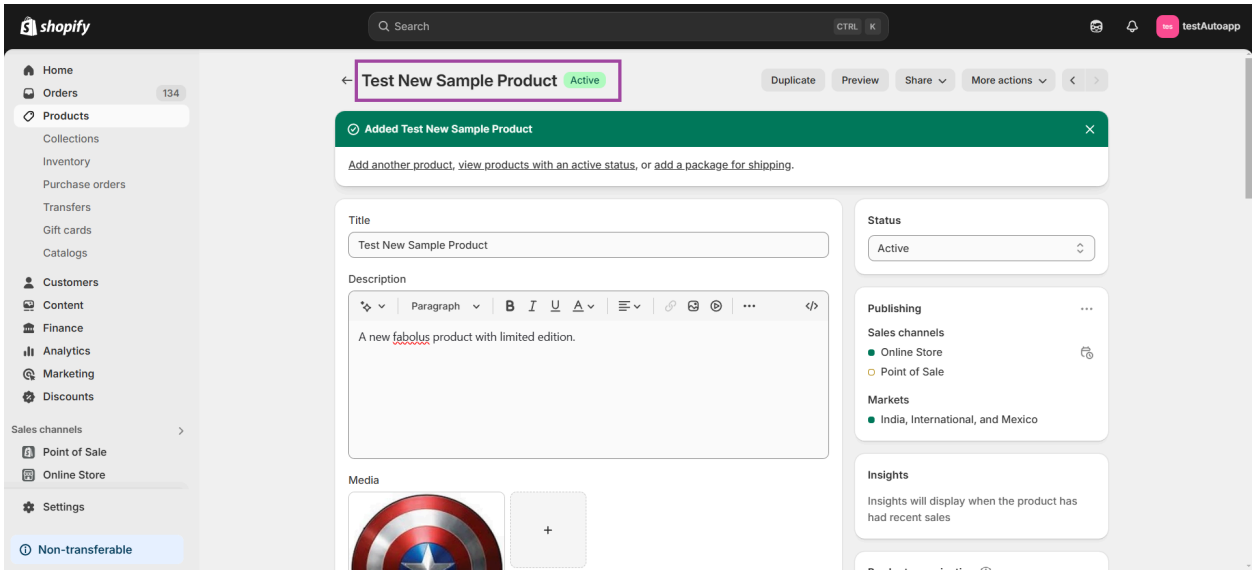
2. **Sync to Salesforce:**
   - The updates made to the record in Shopify will be automatically reflected in Salesforce through the real-time sync process.
3. **Verification:**
   - To verify, navigate to the corresponding object's record page in Salesforce.
   - Check that the updated information matches the changes made in Shopify.

4. Example:

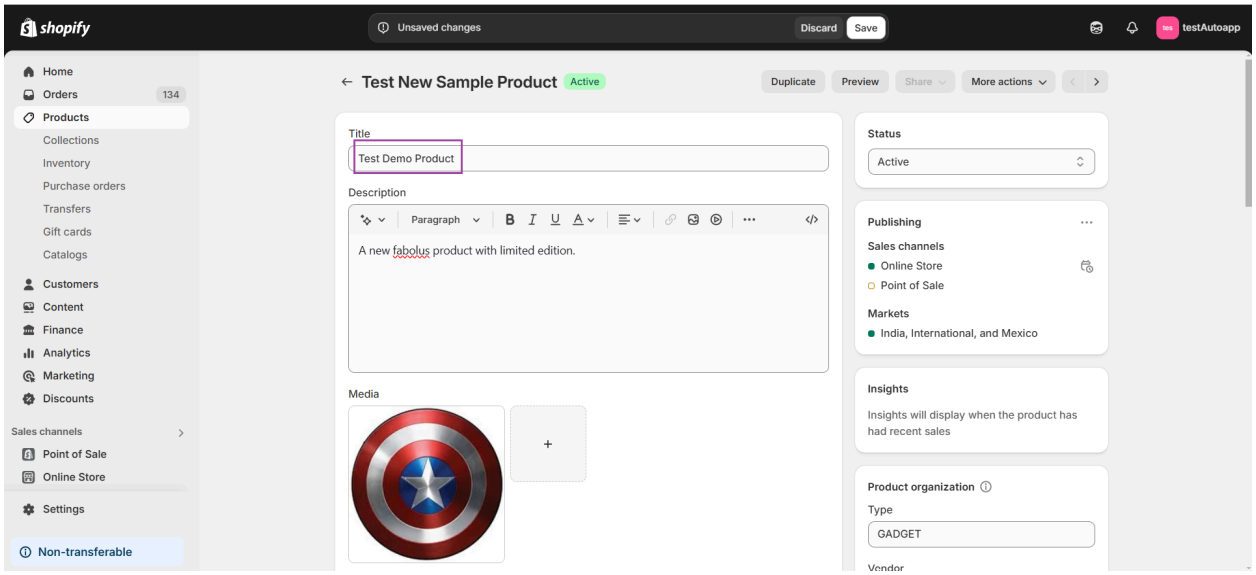
- The following example provides a demonstration of an existing Product update.
  i. Navigate to the Product section in the Shopify Store
  ii. Search for the recently created product or filter it.



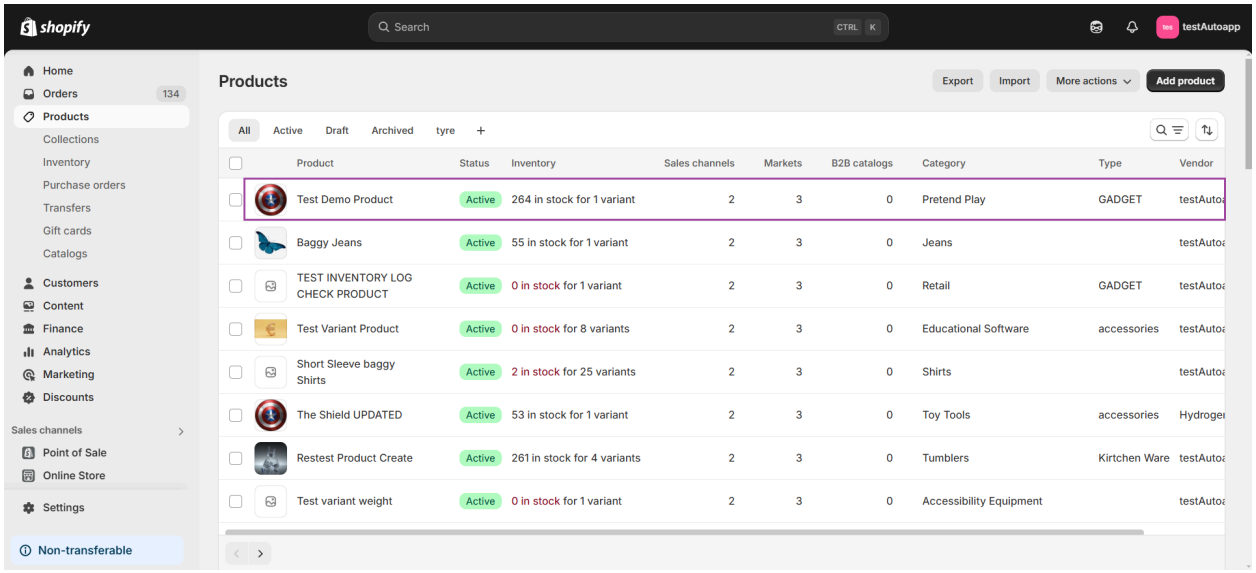
iii. Make sure the product is available in the Salesforce org
iv. Click on the product
v. A page section opens for the existing product.



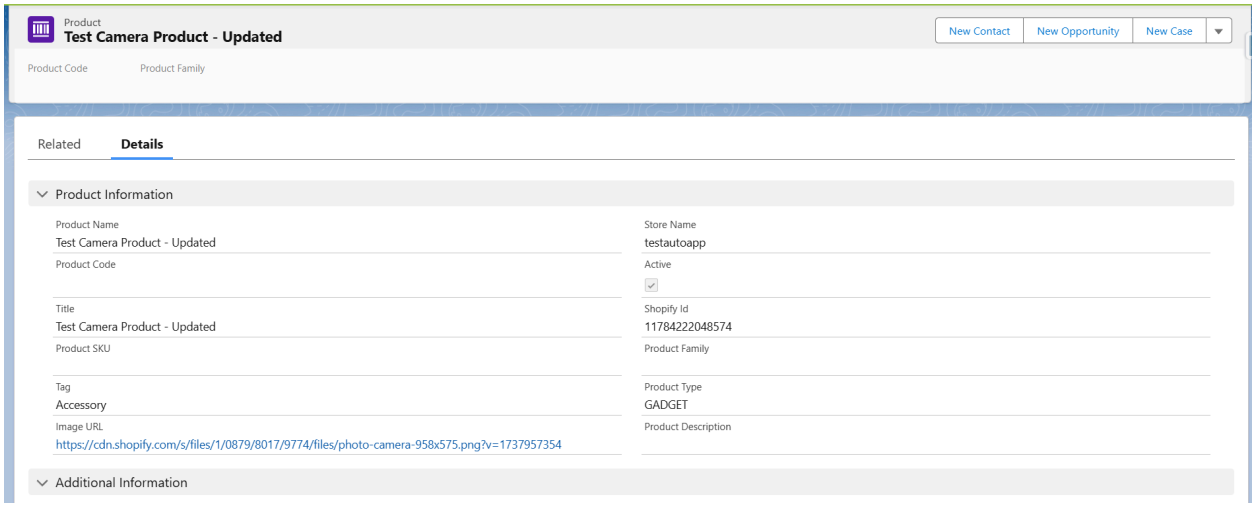
vi. Edit any of the fields. Let's edit the Product name to New Sample Product
    1. Existing Title:- **Test New Sample Product**
    2. Updated Title:- **Test Demo Product**



vii. Click on the **Save** button above the page.
viii. The existing Product has been updated on the Product List.



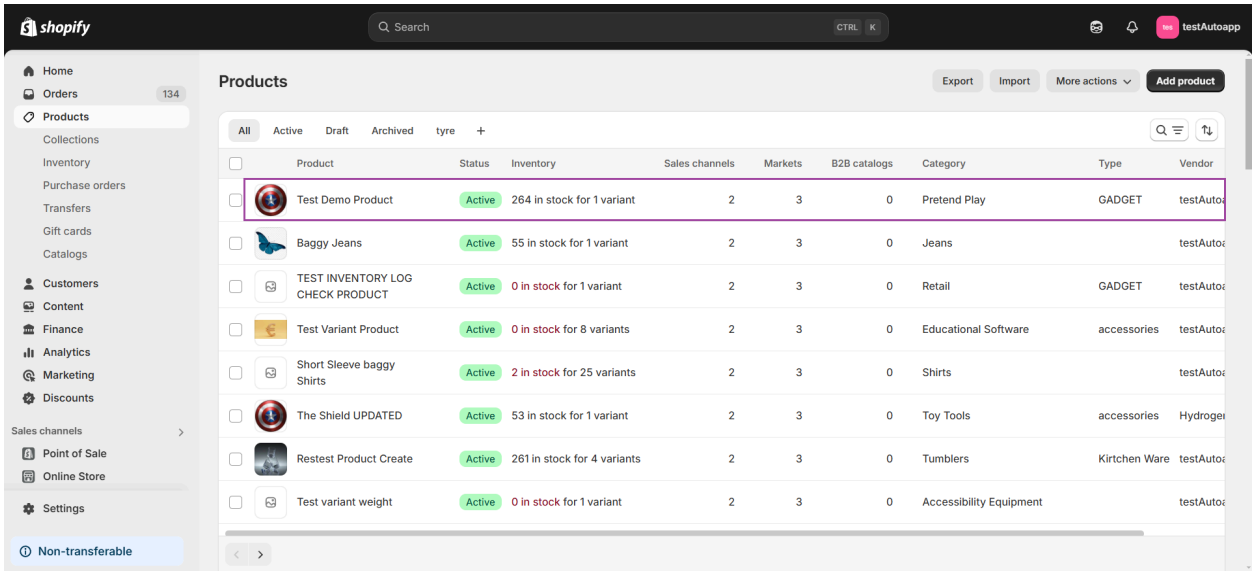
- The same Product with the existing details will be updated in the Salesforce org.



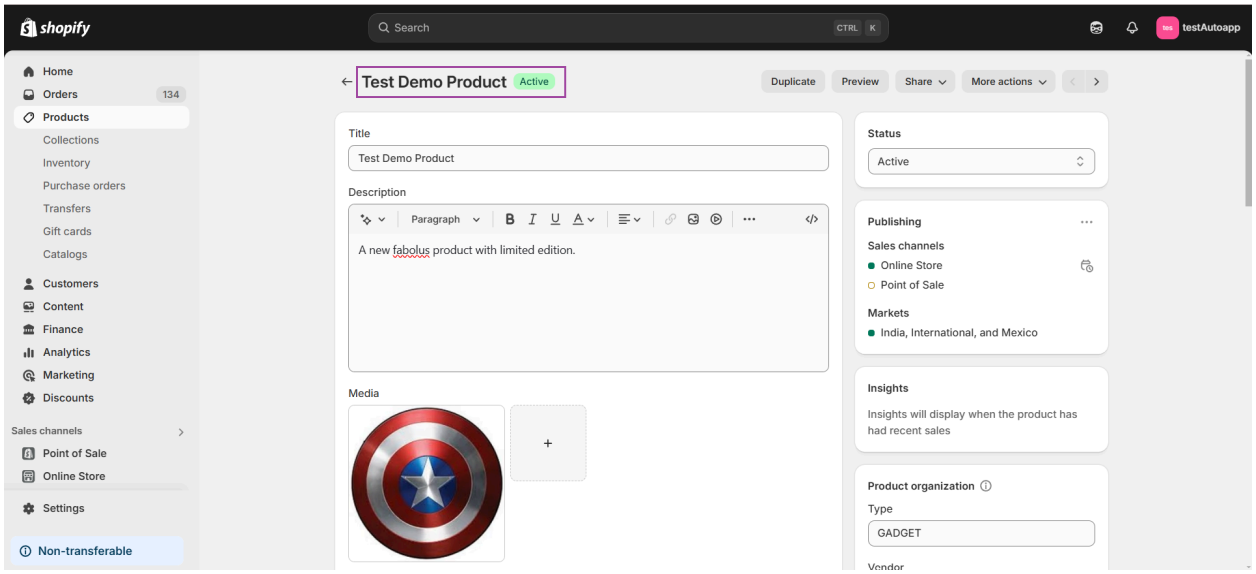
- Verify the field value updation of the existing Product record on the **Product2** record page.
- The same update process applies to the other supported objects for synchronization.

Deletion of Records:

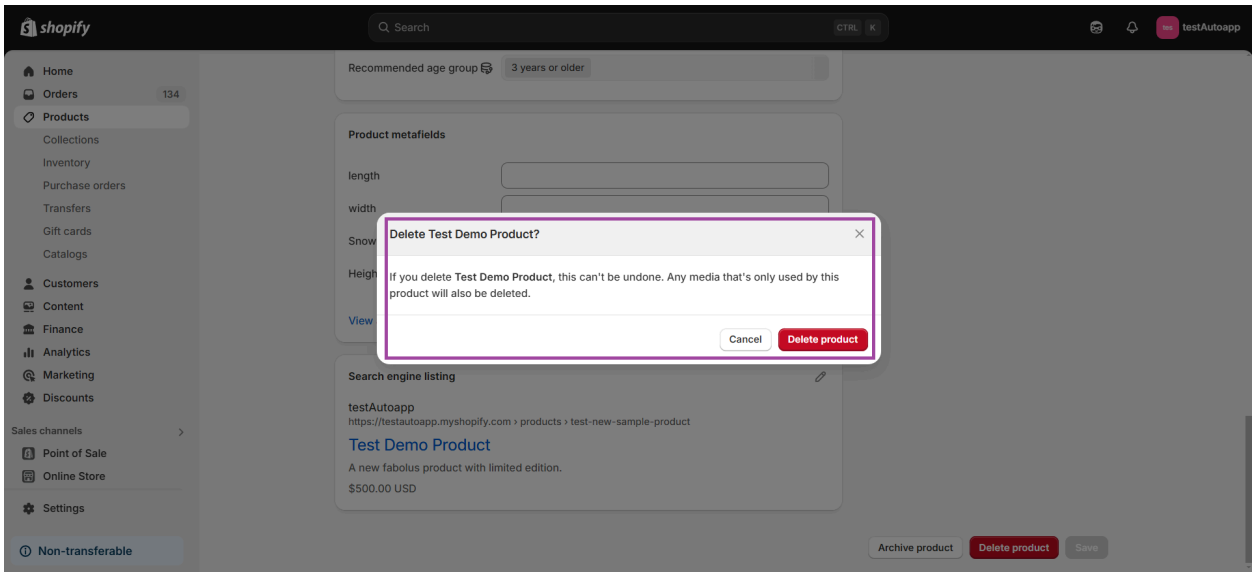
1. **Delete a Record in Shopify:**
    - Navigate to any of the supported objects (e.g., Products, Customers, Orders) in your Shopify store.
    - Select the record you want to delete.
    - Click on the **Delete** option to remove the record.
2. **Sync to Salesforce:**
    - The deletion in Shopify will trigger the deletion process in Salesforce via the real-time sync.
3. **Verification:**
    - To verify, navigate to the corresponding object's record page in Salesforce.
    - Check that the record has been removed from Salesforce, matching the deletion in Shopify.
4. **Example:**
    - The following example provides a demonstration of an existing Product update.
        i. Navigate to the Product section in the Shopify Store
        ii. Search for the recently created product or filter it.



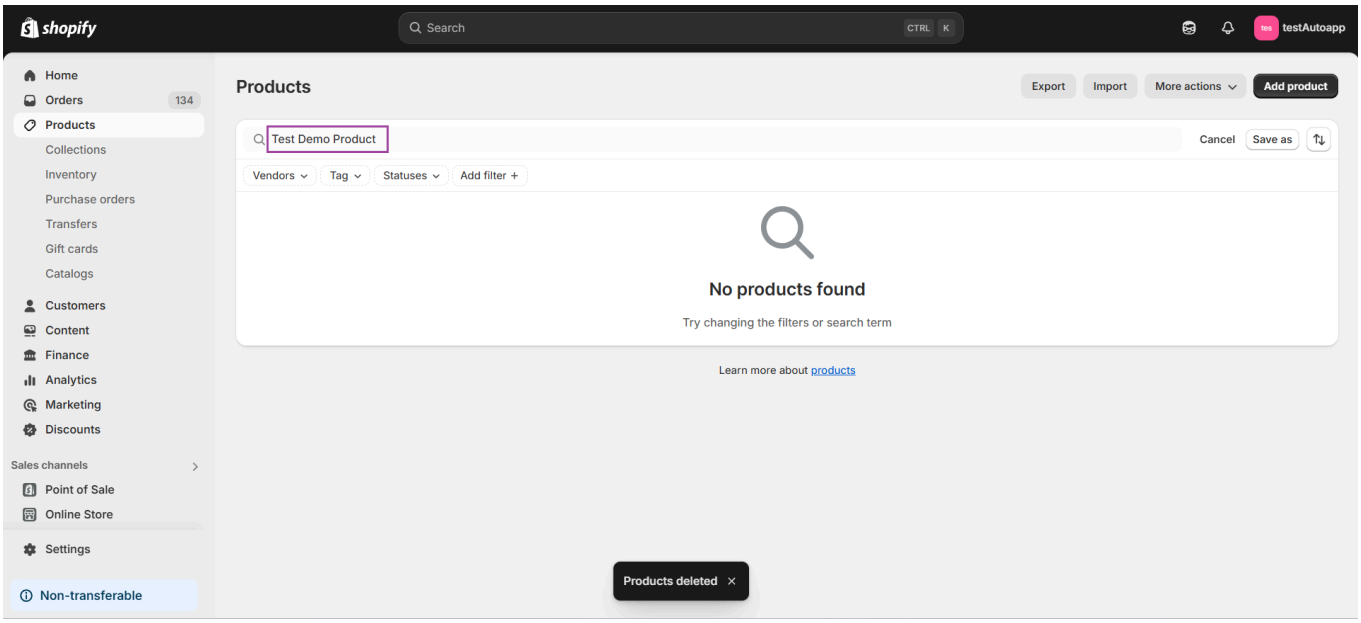
iii. Make sure the product is available in the Salesforce org
iv. Click on the product
v. A page section opens for the existing product.



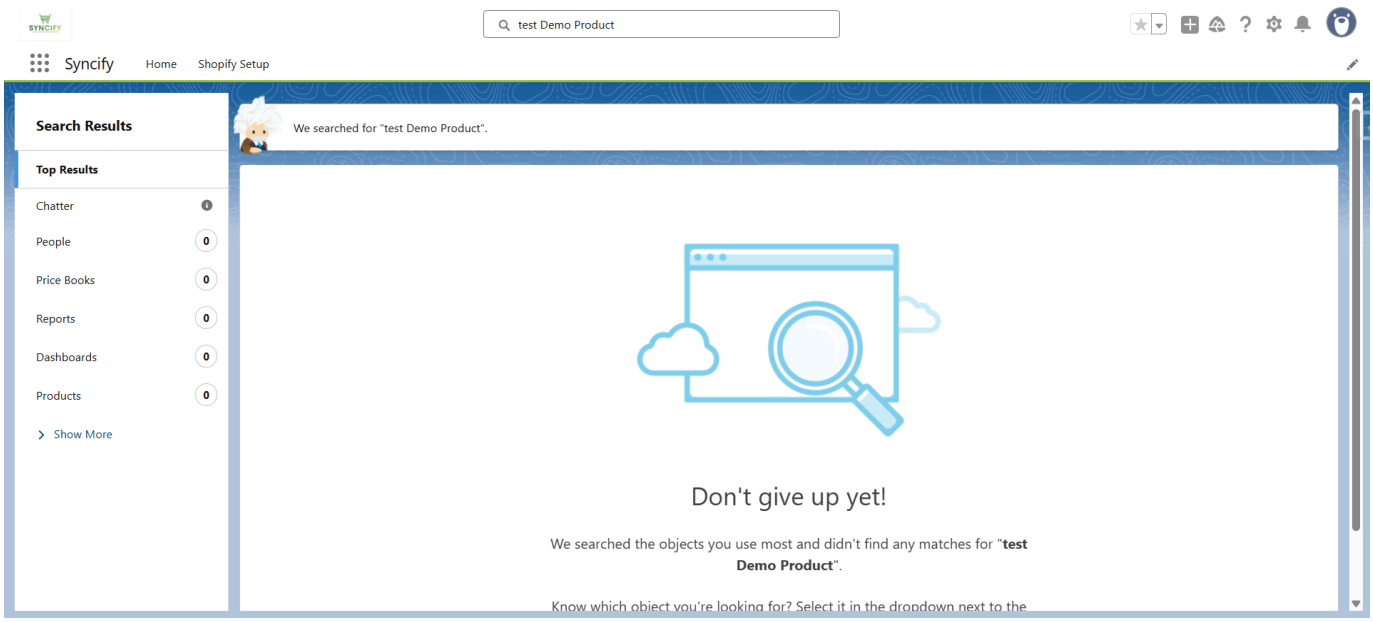
vi. Scroll down to the bottom of the page
vii. Click on the **Delete Product** button
viii. A confirmation tab pops up



ix. Click on the **Delete Product** button
x. Navigate back to the Product list and search for the deleted Product by the title “**New Sample Product**”
xi. The product will no longer appear on the Product list



- The same Product with the existing details will be deleted from the Salesforce org.



- Verify the removal of the existing Product record on the **Product2** record page.
- The same delete process applies to the other supported objects for synchronisation.

Bi-Directional Sync:

Overview:

- The Salesforce–Shopify Bi-Directional Sync enables seamless, near real-time data synchronisation between Salesforce and Shopify. This integration ensures that any changes (updates) made in supported objects in Salesforce are automatically updated in Shopify, keeping both systems perfectly aligned.
- The solution supports synchronisation for key business entities such as Product2, variant_c, Location_c, Inventory_c, InventoryLevel_c, Collection_c, Order, Draft Order, Account and Contact, ensuring that data remains consistent and up to date across both platforms.
- The integration is designed to be flexible and extensible, allowing developers to easily customise or extend the logic through a provided interface class to meet specific business requirements.

Key Highlights:

- **Bi-Directional Real-Time Sync:** Whenever a record is updated in Salesforce, the corresponding record in Shopify is updated instantly.

- **Object & Field-Level Synchronisation:** Only selected objects and fields are synchronised, ensuring optimised performance and controlled data flow between systems.

- **Extensible Interface for Custom Logic:** To simplify customisation, the integration provides a **base interface class** that can be **extended** by developers. Using this approach, teams can add or override sync logic to match unique business rules without altering the core integration framework. (Interface Name: IShopifyInputExtension)

- **Configurable Sync Toggles:** Admin users can enable or disable synchronisation for specific objects directly from the **Shopify Configuration** page in Salesforce.

- **Centralised Error Handling & Logging:** All sync operations are tracked within Salesforce, and any failed sync attempts are logged for review. Automatic and manual retry mechanisms ensure reliability.

Supported Objects and Fields

Product2 → Product

- Supported Shopify Fields:
  - title
  - tag

Variant__c → Product Variants

- Supported Shopify Fields:
  - price

Inventory__c → Inventory Item

- Supported Shopify Fields:
  - tracked

InventoryLevel__c → Inventory Level

- Supported Shopify Fields:
  - available

Location__c → Location

- Supported Shopify Fields:
  - name

Order → Order

- Supported Shopify Fields:
  - note
  - tags

Account / Contact → Customer

- Supported Shopify Fields:
  - last_name
  - phone
  - note
  - tags

DraftOrder__c → Draft Order

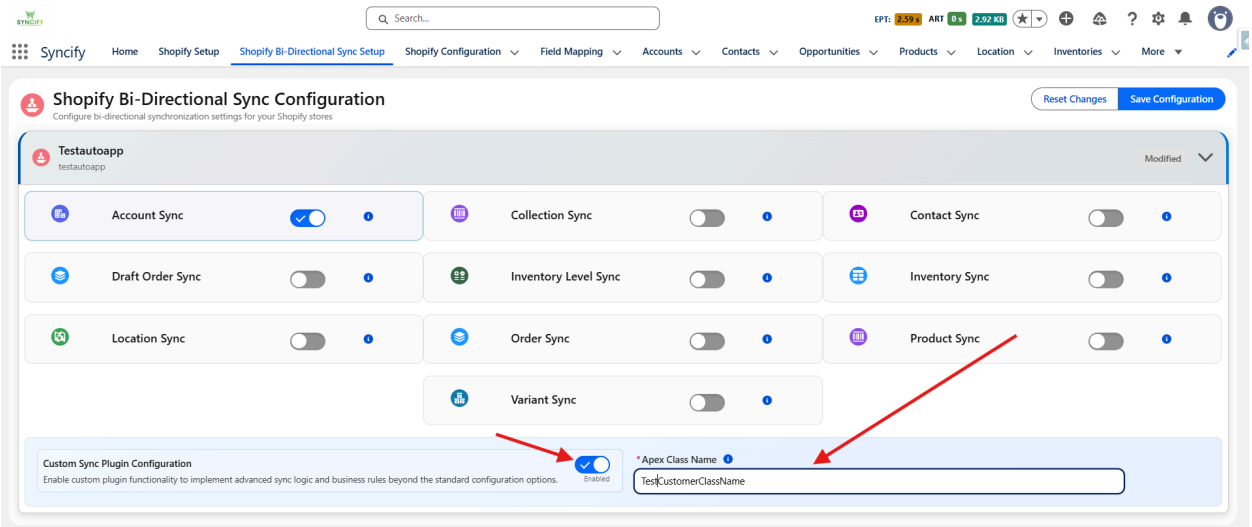
- Supported Shopify Fields:
  - tags

Collection__c → Collection

- **Supported Shopify Fields:**
  - title
  - handle

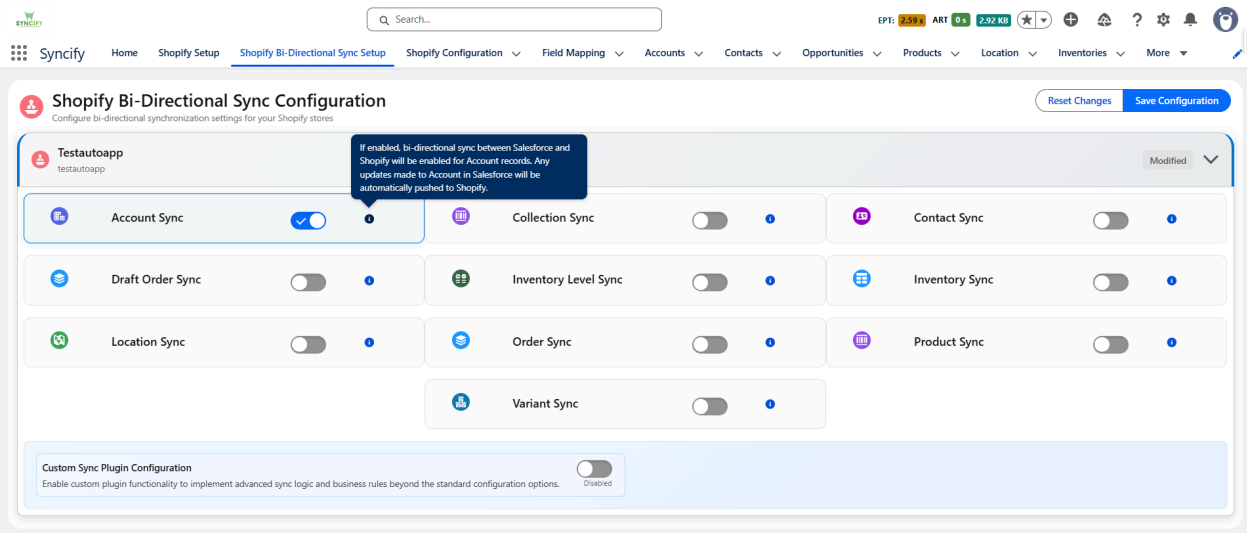
Steps to Implement Custom Logic Extension

1. Create a new Apex class that implements the Icmp.IShopifyInputExtension interface.
2. Define your logic inside the interface methods (for example, modifying field values or applying business rules before syncing).
3. In Salesforce, go to the Shopify Bi-Directional Setup tab.
4. Enable the checkbox “Custom Sync Plugin Configuration.”
5. Enter your custom class name (e.g., *TestCustomerClassName*) in the Custom Class Name field and click Save.
6. Once saved, your custom class will automatically be used during



synchronisation.

7. You can also control each object’s synchronisation by **enabling or disabling it** from the **Shopify Bi-Directional Sync Setup** tab on the Syncify App.



Interface Overview

The **IShopifyInputExtension** interface defines methods for each method that receives:

- inputMap → The data map being sent to Shopify
- record → The Salesforce record of the corresponding object
- fieldMappings → The list of mapped fields between Salesforce and Shopify
- objectType → The object type identifier

Developers can override any of these methods to customise synchronisation logic.

Interface Definition:

```
global class TestCustomerClassName implements lcmp.IShopifyInputExtension {
    global Map<String, Object> enhanceProductInput(Map<String, Object> inputMap, Product2 record, List<lcmp__Field_Mapping__c> fieldMappings, String objectType) {
        // Example: Modify product title before sync
        inputMap.put('title','CusTestCustomerClassName');
        return inputMap;
    }
    global Map<String, Object> enhanceVariantInput(Map<String, Object> inputMap, lcmp__Variant__c record, List<lcmp__Field_Mapping__c> fieldMappings, String objectType) {
        return inputMap;
    }
    global Map<String, Object> enhanceLocationInput(Map<String, Object> inputMap, lcmp__Location__c record, List<lcmp__Field_Mapping__c> fieldMappings, String objectType) {
        return inputMap;
    }
    global Map<String, Object> enhanceInventoryInput(Map<String, Object> inputMap, lcmp__Inventory__c record, List<lcmp__Field_Mapping__c> fieldMappings, String objectType) {
        return inputMap;
    }
    global Map<String, Object> enhanceInventoryLevelInput(Map<String, Object> inputMap, lcmp__Inventory_Level__c record, List<lcmp__Field_Mapping__c> fieldMappings, String objectType) {
        return inputMap;
    }
    global Map<String, Object> enhanceCollectionInput(Map<String, Object> inputMap, lcmp__Collection__c record, List<lcmp__Field_Mapping__c> fieldMappings, String objectType) {
        return inputMap;
    }
    global Map<String, Object> enhanceOrderInput(Map<String, Object> inputMap, Order record, List<lcmp__Field_Mapping__c> fieldMappings, String objectType) {
        return inputMap;
    }
    global Map<String, Object> enhanceDraftOrderInput(Map<String, Object> inputMap, Order record, List<lcmp__Field_Mapping__c> fieldMappings, String objectType) {
        return inputMap;
    }
    global Map<String, Object> enhanceAccountInput(Map<String, Object> inputMap, Account record, List<lcmp__Field_Mapping__c> fieldMappings, String objectType) {
        return inputMap;
    }
    global Map<String, Object> enhanceContactInput(Map<String, Object> inputMap, Contact record, List<lcmp__Field_Mapping__c> fieldMappings, String objectType) {
        return inputMap;
    }
}
```

Conclusion:

- The Syncify App for Salesforce delivers a comprehensive and intuitive solution for real-time data synchronisation between Shopify and Salesforce. By supporting seamless integration for key objects such as Products, Orders, and Customers, and providing features like real-time sync, customizable field mapping, and batch import functionality, Syncify ensures consistent and accurate data across both platforms.

- The app's guided configuration process enables users to easily set up store credentials, define object mappings, and manage synchronisation workflows without complexity. Its flexible architecture supports fine-grained control through object toggles and custom field mappings, allowing organisations to tailor data flow according to their business requirements.

- With built-in capabilities to track sync progress, monitor record creation, updates, and deletions, and handle multi-store environments, Syncify enhances operational efficiency and simplifies cross-platform data management.

- Overall, the Syncify App provides a robust and scalable bridge between Shopify and Salesforce, ensuring seamless integration, reliable data consistency, and up-to-date insights — empowering businesses to streamline operations and make data-driven decisions with confidence.