

# Phase 5

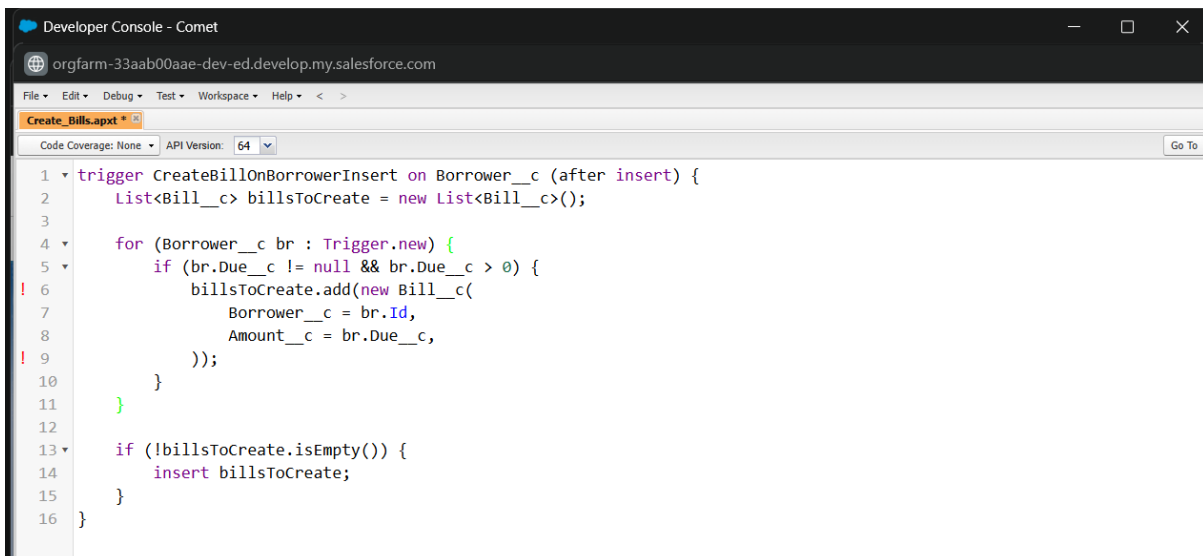
This phase discusses about the use of apex programming/automation in the project

## SHELF SYNC LIBRARY

❖ **Apex Triggers** : Apex triggers can be used to automate the task which gets triggered :

- before insert
- after insert
- before update
- after update
- before delete
- after delete
- after undelete

### 1. Create Bills Apex Trigger (after insert)



The screenshot shows the Salesforce Developer Console interface. The browser address bar displays 'orgfarm-33aab00aae-dev-ed.develop.my.salesforce.com'. The editor window is titled 'Create\_Bills.apxt' and shows the following Apex code:

```
1 trigger CreateBillOnBorrowerInsert on Borrower__c (after insert) {
2     List<Bill__c> billsToCreate = new List<Bill__c>();
3
4     for (Borrower__c br : Trigger.new) {
5         if (br.Due__c != null && br.Due__c > 0) {
6             billsToCreate.add(new Bill__c(
7                 Borrower__c = br.Id,
8                 Amount__c = br.Due__c,
9             ));
10        }
11    }
12
13    if (!billsToCreate.isEmpty()) {
14        insert billsToCreate;
15    }
16 }
```

Is an example of a apex trigger which could be deployed to create the bill records as soon as the borrowers due increases or is greater than 0.

2. To reduce the stock amount of the books as soon as the borrower record is created which means member issues a copy of the book from the available stock.

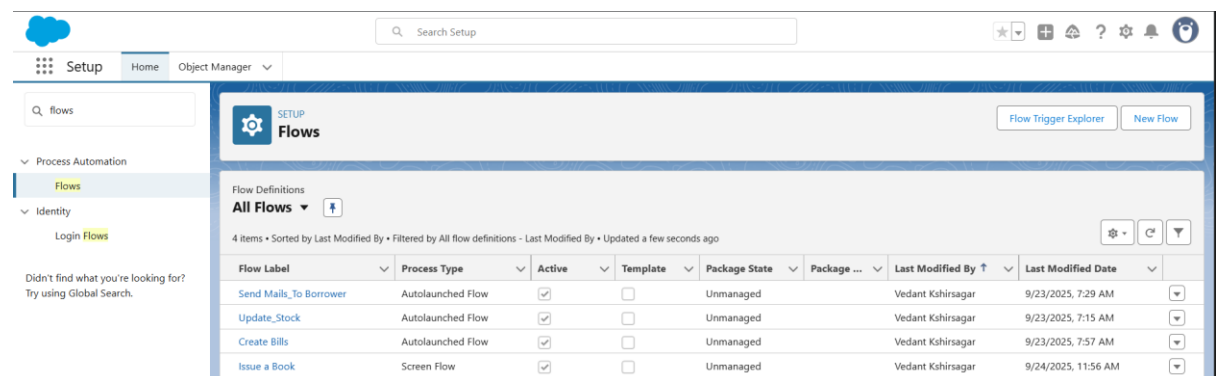
```
1 trigger DecreaseBookStockOnBorrowerInsert on Borrower__c (after insert) {
2     Set<Id> bookIds = new Set<Id>();
3     for (Borrower__c br : Trigger.new) {
4         if (br.Book__c != null) {
5             bookIds.add(br.Book__c);
6         }
7     }
8
9     if (!bookIds.isEmpty()) {
10         List<Book__c> booksToUpdate = [SELECT Id, Stock__c FROM Book__c WHERE Id IN :bookIds];
11         Map<Id, Book__c> bookMap = new Map<Id, Book__c>();
12         for (Book__c bk : booksToUpdate) {
13             bookMap.put(bk.Id, bk);
14         }
15
16         for (Borrower__c br : Trigger.new) {
17             if (br.Book__c != null && bookMap.containsKey(br.Book__c)) {
18                 Book__c bookRec = bookMap.get(br.Book__c);
19                 if (bookRec.Stock__c != null && bookRec.Stock__c > 0) {
20                     bookRec.Stock__c -= 1;
21                 }
22             }
23         }
24
25         update booksToUpdate;
26     }
27 }
28
29
```

The above can be an example code for it.

In this project to create records or send emails or update certain records a complex code or trigger is not needed

Instead of complex triggers I have used flows which automate the task without the use of codes

Flows created are :



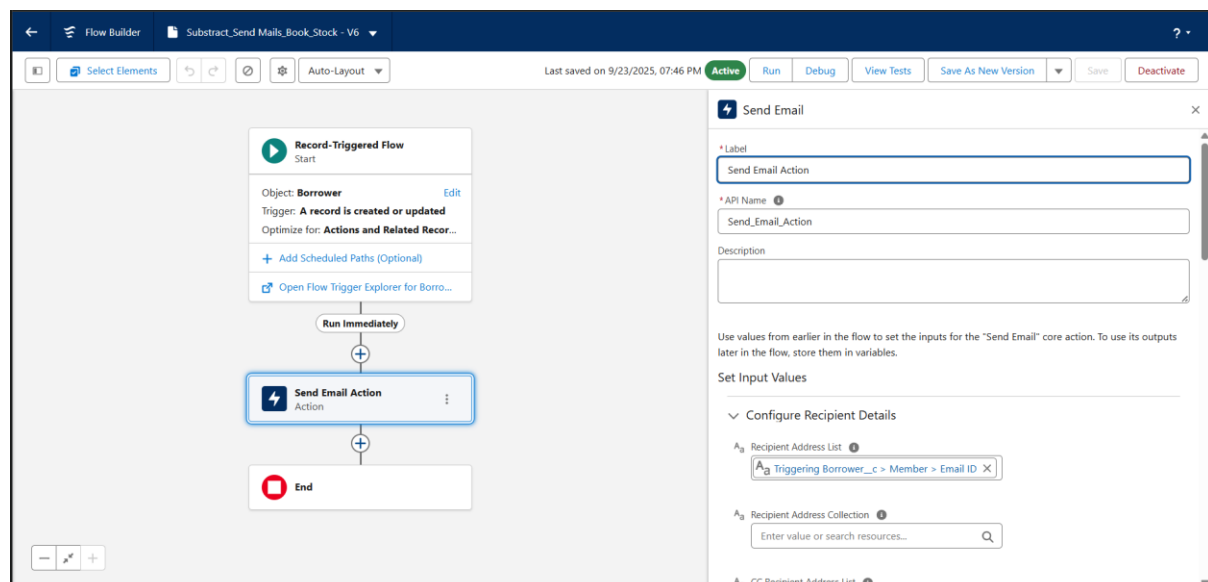
The screenshot shows the Salesforce Setup interface with the 'Flows' section selected. A table lists the following flows:

Flow Label	Process Type	Active	Template	Package State	Package ...	Last Modified By	Last Modified Date
Send Mails_To Borrower	Autolaunched Flow	✓	□	Unmanaged		Vedant Kshirsagar	9/23/2025, 7:29 AM
Update_Stock	Autolaunched Flow	✓	□	Unmanaged		Vedant Kshirsagar	9/23/2025, 7:15 AM
Create Bills	Autolaunched Flow	✓	□	Unmanaged		Vedant Kshirsagar	9/23/2025, 7:57 AM
Issue a Book	Screen Flow	✓	□	Unmanaged		Vedant Kshirsagar	9/24/2025, 11:56 AM

- Send\_Mails\_To Borrowers
- Update\_Stock
- Create Bills
- Issue a Book

These flows perform the complex task instead of an Apex Program

## 1. Sending Email Confirmation to the Borrowers.



The mail Subject is as follows where the “{!\$Record.Book\_\_r.Name}” fetches the books name that the borrower issued here.

Edit Text Template

\* API Name ⓘ

Mail\_Subject

Description

Subject\_Lib

\* Body ⓘ

Insert a resource... 🔍

View as Plain Text ▼

{!\$Record.Book\_\_r.Name} Borrow Confirmation

Cancel

Done

The body of the email is :

Edit Text Template

\* API Name ⓘ

Mail

Description

Send mail as a vote of thanks and return alert to all borrowers.

\* Body ⓘ

Insert a resource... 🔍

View as Plain Text ▼

Hey {!\$Record.Member\_\_r.Name},  
Thank you for borrowing a book from our collection. We hope you find it both enjoyable and insightful. Please kindly note that the

Cancel

Done

Hey {!\$Record.Member\_\_r.Name},

Thank you for borrowing a book from our collection. We hope you find it both enjoyable and insightful. Please kindly note that the book should be returned

within 10 days from the borrowing date. If the book is not returned by then, a fine of \$50 per day will be applied. We appreciate your understanding and cooperation. Should you have any questions or need an extension, please feel free to contact us.

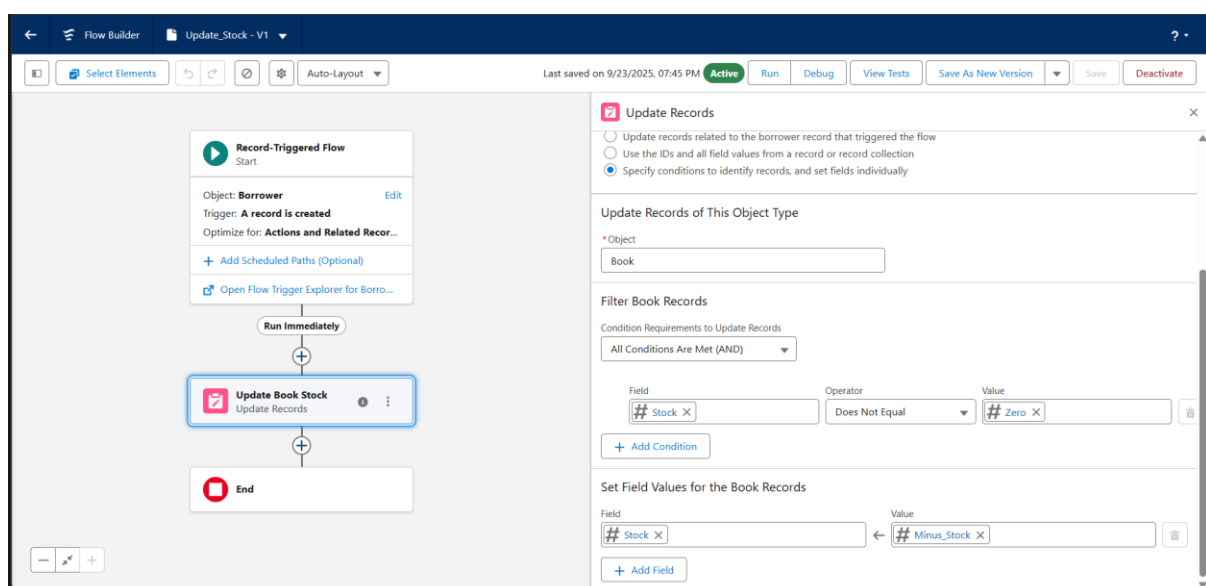
Best regards,

{!\$User.FirstName} {!\$User.LastName}

Your Return Date is : {!\$Record.Return\_Date\_\_c}

This represents the body which fetches the details like the member name the username who is sending the mail and the return date for the respective book.

## 2. Update\_Stock



This flow updates the stock to stock – 1 in the book records for the same book when the borrower issues the book.



The Minus\_Stock Resource that is used here is :

Edit Formula

---

\* API Name ⓘ  
Minus\_Stock

Description

\* Data Type ⓘ  
Number ▼

Decimal Places  
0

\* Formula

Insert a resource... 🔍

All Functions ▼

Insert a function... 🔍

Select an Operator... ▼

`{!$Record.Book__r.Stock__c}-1`

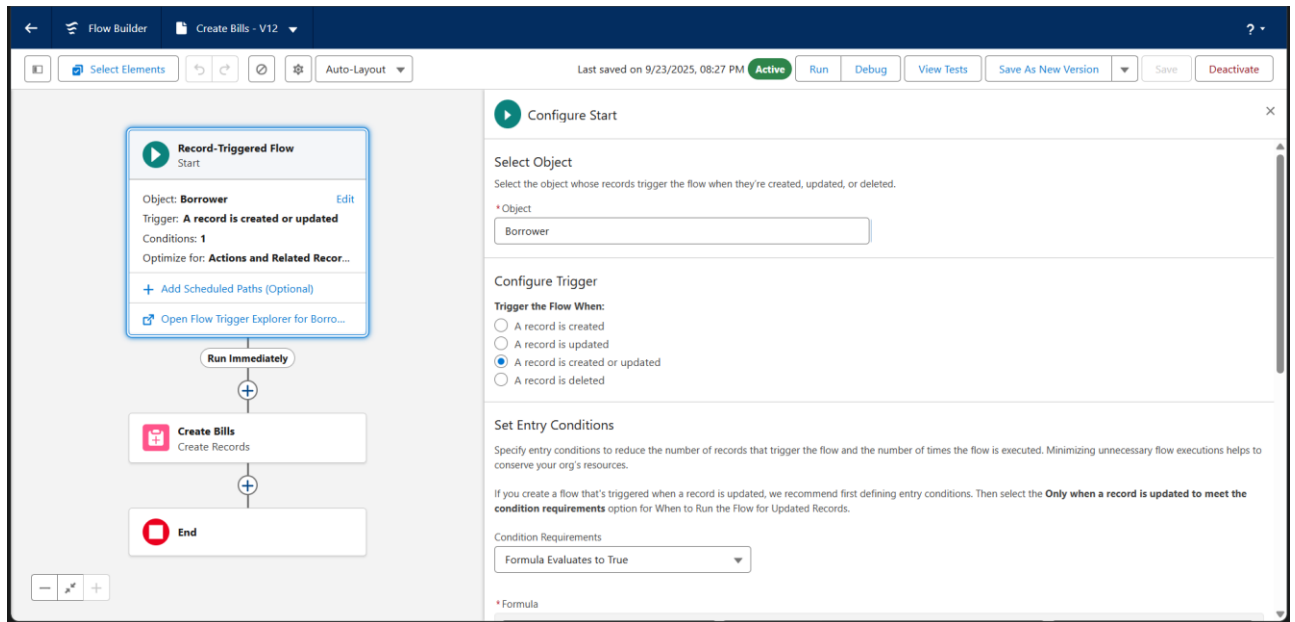
Check Syntax

Cancel Done

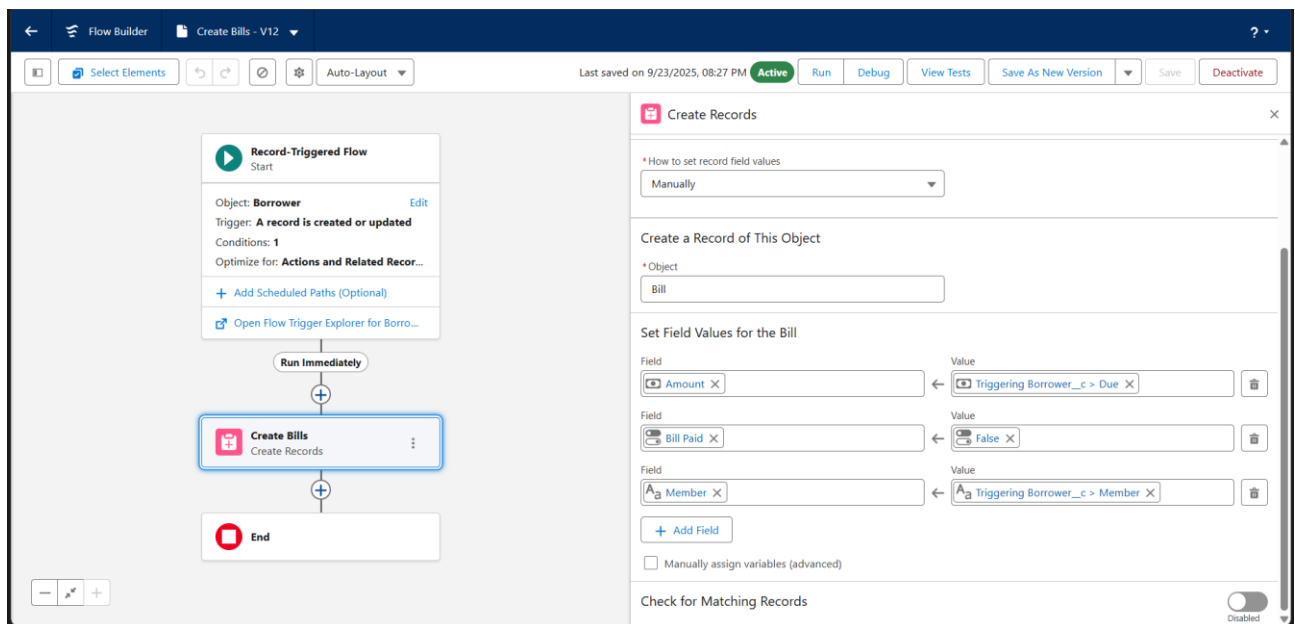
The formulae : `{!$Record.Book__r.Stock__c}-1`

Is used to reduce the stock and update it in the record respectively.

### 3. Create Bills :



This is a triggered flow which triggers when a borrower record is created and the due is over 50 \$.



This helps create the bills for the member whose book is over the return date.