Phase 5 Apex Programming

Comprehensive documentation of Apex Programming development tasks completed for the Pharmacy Inventory System project, featuring advanced validation logic and seamless user experience integration.

Data Model Foundation

Formula Field Creation

A critical **Price__c** formula field was established on the Prescription object to automatically calculate total costs by multiplying quantity dispensed by product price.

Navigation Path:
Setup > Object Manager > Prescription > Fields & Relationships

Formula Logic

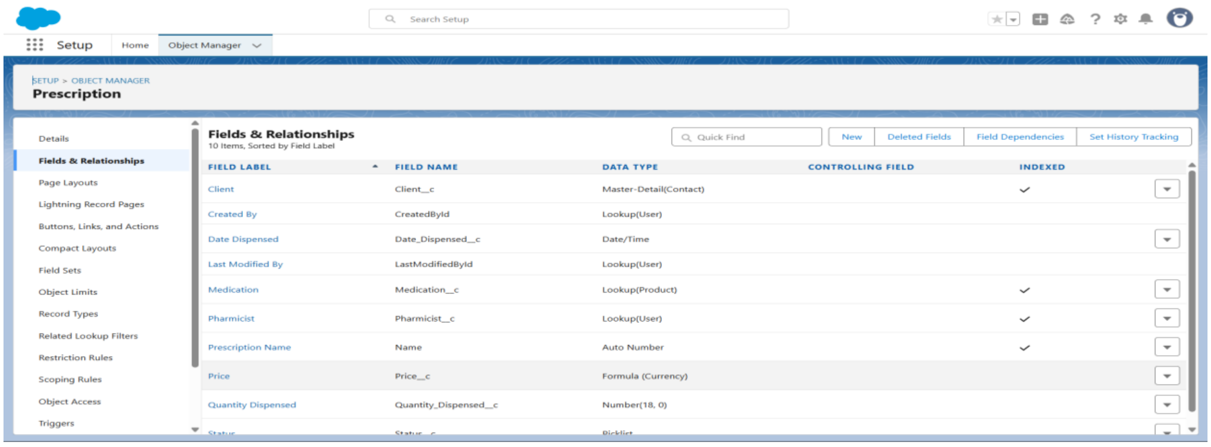
Quantity_Dispensed__c × Price__c from related Product record

Object Target

Prescription object with automatic calculation

Business Impact

Real-time cost tracking and inventory valuation



Prescription

Fields & Relationships
10 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Client	Client__c	Master-Detail(Contact)		✓
Created By	CreatedById	Lookup(User)		
Date Dispensed	Date_Dispensed__c	Date/Time		
Last Modified By	LastModifiedById	Lookup(User)		
Medication	Medication__c	Lookup(Product)		✓
Pharmicist	Pharmicist__c	Lookup(User)		✓
Prescription Name	Name	Auto Number		✓
Price	Price__c	Formula (Currency)		
Quantity Dispensed	Quantity_Dispensed__c	Number(18, 0)		
Status	Status__c	Picklist		

Professional Development Architecture

Project Structure Setup

Established VS Code project with proper folder organization following Salesforce DX standards and professional development practices.

Trigger Handler Pattern

Implemented industry-standard Trigger Handler pattern for maintainable, scalable, and testable Apex code architecture.

SOQL Integration

Developed efficient SOQL queries to retrieve and validate inventory data against prescription requirements in real-time.

Core Apex Components

PrescriptionTriggerHandler.cls

Central handler class containing the **validateStockAvailability** method with sophisticated inventory validation logic.

- SOQL query for Quantity_on_Hand__c retrieval
- Comparison logic against Quantity_Dispensed__c
- Error blocking via addError() method

PrescriptionTrigger.trigger

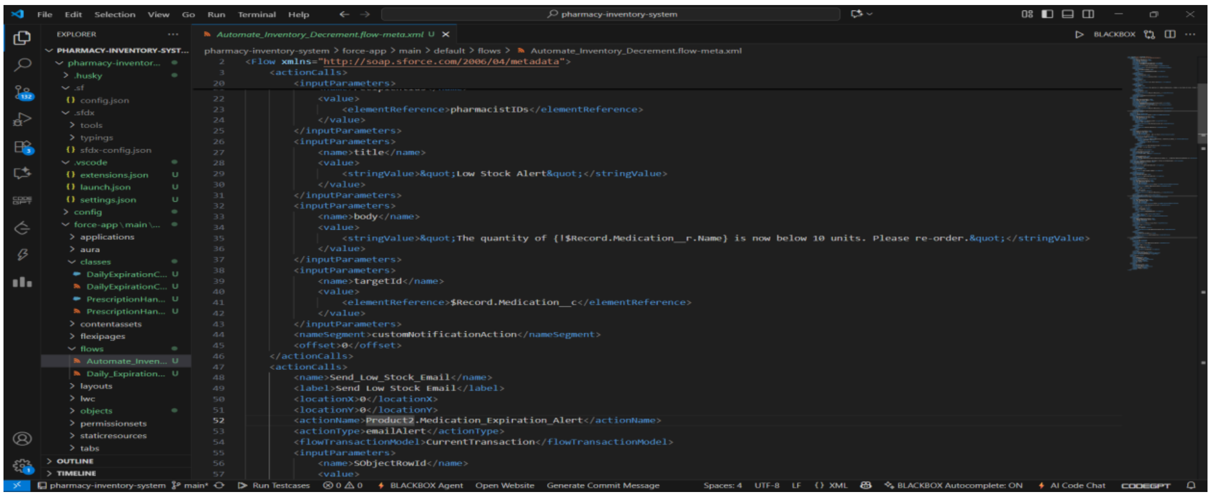
Streamlined trigger implementation firing on **before insert** and **before update** events.

- Clean separation of concerns
- Handler class method invocation
- Event-driven validation execution

PrescriptionTriggerTest.cls

Comprehensive test coverage with positive and negative test scenarios ensuring robust validation.

- Test data creation (Product, Contact, User)
- Valid prescription insertion testing
- Error handling verification with try/catch blocks



pharmacy-inventory-system > force-app > main > default > flows > Automate_Inventory_Decrement.flow-meta.xml

```xml
<Flow xmlns="http://soap.sforce.com/2006/04/metadata">
    <actionCalls>
        <inputParameters>
            <value>
                <elementReference>pharmacistIDs</elementReference>
            </value>
        </inputParameters>
        <inputParameters>
            <name>title</name>
            <value>
                <stringValue>&quot;Low Stock Alert&quot;</stringValue>
            </value>
        </inputParameters>
        <inputParameters>
            <name>body</name>
            <value>
                <stringValue>&quot;The quantity of {!$Record.Medication__r.Name} is now below 10 units. Please re-order.&quot;</stringValue>
            </value>
        </inputParameters>
        <inputParameters>
            <name>targetId</name>
            <value>
                <elementReference>$Record.Medication__c</elementReference>
            </value>
        </inputParameters>
        <nameSegment>customNotificationAction</nameSegment>
        <offset>0</offset>
    </actionCalls>
    <actionCalls>
        <name>Send_Low_Stock_Email</name>
        <label>Send Low Stock Email</label>
        <locationX>0</locationX>
        <locationY>0</locationY>
        <actionName>Product2.Medication_Expiration_Alert</actionName>
        <actionType>emailAlert</actionType>
        <flowTransactionModel>CurrentTransaction</flowTransactionModel>
        <inputParameters>
            <name>SObjectRowId</name>
            <value>
```

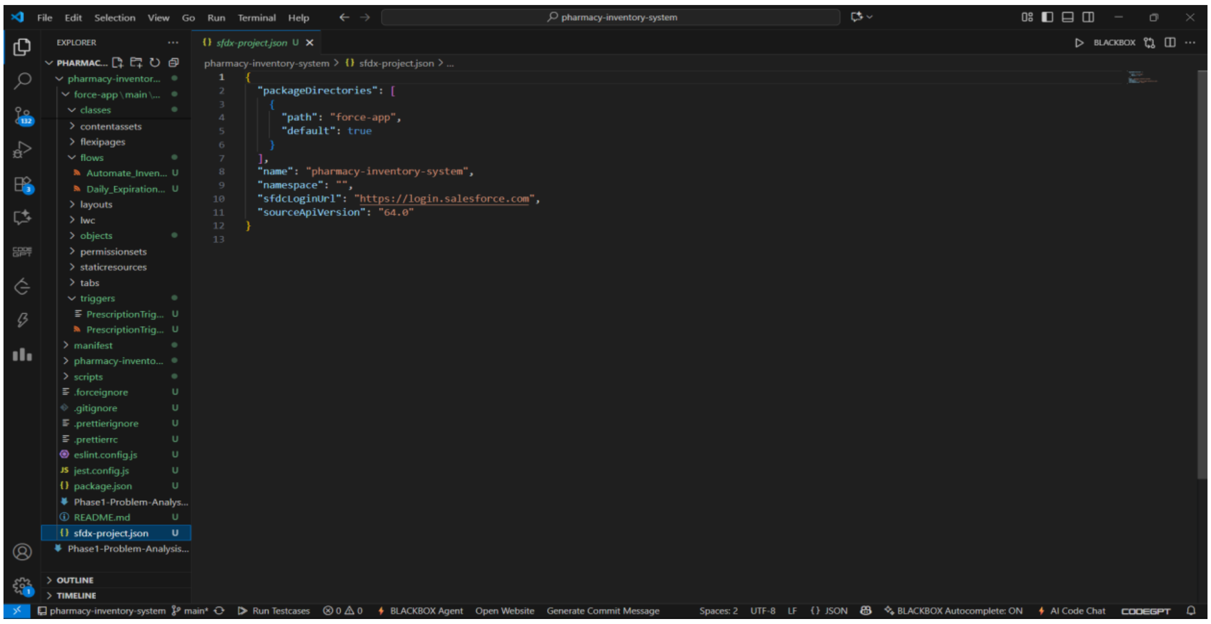
Deployment & Quality Assurance

Deployment Process

Utilized **SF: Deploy Source to Org** command from VS Code for seamless integration with the Salesforce environment.

Debugging Excellence

- Corrected custom field API name inconsistencies
- Resolved Apex syntax errors through systematic review
- Validated deployment success across all components


```json
{
  "packageDirectories": [
    {
      "path": "force-app",
      "default": true
    }
  ],
  "name": "pharmacy-inventory-system",
  "namespace": "",
  "sfdcLoginUrl": "https://login.salesforce.com",
  "sourceApiVersion": "64.0"
}
```

Enhanced User Experience Integration

Error Screen Creation

Added dedicated error screen to the **Dispense Medication Wizard** Screen Flow for professional error handling and user guidance.

Dynamic Error Display

Configured Display Text component to show **{!$Flow.FaultMessage}** global variable, automatically capturing Apex trigger errors.

Fault Path Implementation

Established red dotted fault path connecting Create Records element to Error Screen, ensuring seamless error redirection.

Comprehensive Testing Strategy

User Authentication

Logged in as Pharmacist user to simulate authentic real-world usage scenarios and validate role-based functionality.

Validation Testing

Verified that Dispense Medication Wizard correctly blocks prescriptions exceeding available inventory quantities.

Error Message Verification

Confirmed custom Apex trigger error messages display properly on the flow's Error Screen interface.

End-to-End Validation

Complete workflow testing from prescription creation through error handling and user feedback mechanisms.

Version Control & Code Management

Git Workflow Implementation

Following successful deployment and comprehensive testing, all Apex code files were systematically committed to the project's GitHub repository.

Standard Git Commands

```
git add .

git commit -m "Phase 5: Apex Programming Complete"

git push origin main
```

This ensures **version tracking**, **code backup**, and **team collaboration** capabilities.

Code Files Saved

.cls and .trigger files securely stored

Backup Security

Repository-based code protection

Team Access

Collaborative development enabled