# Pharmacy Delivery CRM on Salesforce
# Project Documentation

## 1. Introduction

This document describes the design and implementation of a Pharmacy Delivery Customer Relationship Management (CRM) system built on Salesforce. The solution is intentionally scoped as a minimum viable product (MVP) that can be implemented in a single day by a beginner Salesforce admin, while still reflecting realistic pharmacy delivery operations.

## 2. Problem Statement

Retail and online pharmacies face operational challenges in managing home deliveries of medications to patients. Typical issues include manual tracking of deliveries, lack of visibility into delivery status, data entry errors, and poor reporting. There is a need for a simple CRM-based solution to track deliveries, enforce basic business rules, and provide operational visibility.

## 3. Objectives

- Centralize information about pharmacies, patients, and deliveries.
- Enforce simple business rules such as preventing past-dated deliveries and missing addresses.
- Automate population of delivery address from patient records to reduce manual errors.
- Provide basic reporting on delivery status for operational monitoring.

## 4. Scope

In scope:
- Use of standard Salesforce objects: Account (Pharmacy) and Contact (Patient).
- Creation of a custom Delivery object to track deliveries.
- Two validation rules on the Delivery object.
- One record-triggered Flow to auto-populate the delivery address.
- One summary report of deliveries by status.

Out of scope:
- Full prescription management and medication line items.
- Integration with external pharmacy or logistics systems.
- Advanced security or mobile app configuration.

## 5. High-Level Design

The solution reuses Salesforce standard objects for core entities:
- Account: represents pharmacies.
- Contact: represents patients.

A custom object Delivery__c is introduced to represent individual delivery orders. Delivery__c is related to Contact via a lookup field (Patient__c) and to Account via a lookup field (Pharmacy__c). The object also stores status, delivery date, delivery address, payment method, and amount.

## 6. Data Model Details

Standard Objects:
1) Account
   - Used to represent pharmacies.

2) Contact
   - Used to represent patients.
   - Key fields used by the solution include Name, Phone, Email, and Mailing Address.

Custom Object:
3) Delivery__c
   - Delivery Number (Auto Number)
   - Patient__c (Lookup to Contact)
   - Pharmacy__c (Lookup to Account)
   - Status__c (Picklist: New, Scheduled, Out for Delivery, Delivered, Failed, Cancelled)

- Delivery_Date__c (Date)
- Delivery_Address__c (Long Text Area)
- Payment_Method__c (Picklist: Cash on Delivery, Card, Online Prepaid)
- Amount__c (Currency)

## 7. Business Rules (Validation Rules)

Two validation rules are implemented on Delivery__c:

Rule 1: Delivery date cannot be in the past
Formula:

```
AND(
  NOT(ISBLANK(Delivery_Date__c)),
  Delivery_Date__c < TODAY()
)
```

Description: Prevents users from scheduling deliveries with a date before today.

Rule 2: Address required when status is Delivered
Formula:

```
AND(
  ISPICKVAL(Status__c, "Delivered"),
  ISBLANK(Delivery_Address__c)
)
```

Description: Ensures that completed deliveries always have a delivery address recorded.

## 8. Automation Design (Flow)

A record-triggered Flow is used to automatically populate the delivery address from the patient (Contact) when a Delivery is created.

Trigger:
- Object: Delivery__c
- Event: When a record is created

Logic:
- Get the Contact record referenced by Patient__c.
- Build an address string from the patient's mailing address fields (e.g., street, city, state, postal code).
- Update the Delivery record's Delivery_Address__c field with this address.

This automation reduces manual data entry and the risk of transcription errors.

## 9. Reporting

A summary report is created to provide operational visibility:

Report: Deliveries by Status
- Report Type: Deliveries
- Grouped by: Status__c
- Shows the count of deliveries per status for a chosen date range.

This gives stakeholders a quick view of the distribution of New, Scheduled, Out for Delivery, Delivered, and Failed deliveries.

## 10. UML Class Diagram (Text Representation)

The following PlantUML-style class diagram represents the core data model:

```
@startuml
class Account {
  +Name
}

class Contact {
```

```
  +FirstName
  +LastName
  +Email
  +Phone
  +MailingStreet
  +MailingCity
  +MailingState
  +MailingPostalCode
}

class Delivery__c {
  +Delivery_Number__c
  +Delivery_Date__c
  +Status__c
  +Delivery_Address__c
  +Payment_Method__c
  +Amount__c
}

Account "1" -- "*" Delivery__c : Pharmacy__c
Contact "1" -- "*" Delivery__c : Patient__c
@enduml
```

This diagram shows that one Account (pharmacy) and one Contact (patient) can each be associated with many Delivery__c records.

## 11. Testing Approach

Basic test scenarios executed for the MVP include:
- Creating a patient (Contact) with a valid mailing address.
- Creating a pharmacy (Account).
- Creating a Delivery with a future Delivery Date and verifying that the Flow copies the address from the patient.
- Attempting to save a Delivery with a past Delivery Date and confirming the validation error.
- Attempting to set Status__c to Delivered without a Delivery_Address__c and confirming the validation error.
- Running the Deliveries by Status report and confirming correct grouping and counts.

## 12. Conclusion

The Pharmacy Delivery CRM built on Salesforce provides a simple yet realistic implementation of a delivery tracking system. It demonstrates core CRM concepts such as data modeling, validation, automation with Flow, and reporting. The solution is suitable as an academic project or entry-level portfolio piece and can be expanded in the future to handle prescriptions, medications, advanced security, and integrations.