Intelligent Last Mile Delivery Orchestration System

# Phase 1: Problem Understanding & Industry Analysis

## Problem Statement

In today’s competitive logistics environment, last-mile delivery remains the most challenging and costly part of the supply chain. Many organizations face issues such as inefficient route planning, delayed deliveries, lack of real-time tracking, and poor customer communication. This results in increased operational costs, reduced customer satisfaction, and limited visibility into delivery status. The objective of this Salesforce project is to design and implement an Intelligent Last Mile Delivery Orchestration system that leverages Salesforce CRM and real-time data to automate delivery scheduling, optimize routes dynamically, monitor live deliveries, and enhance customer communication, thereby improving operational efficiency and customer experience.

## Objectives

The main objectives of this phase are:  
- To analyze the logistics industry and identify the pain points in last-mile delivery.  
- To understand how Salesforce CRM and integrations can help address these challenges.  
- To define the high-level goals of the Intelligent Delivery Orchestration System.

## Activities Done

- Conducted research on last-mile delivery challenges faced by logistics companies.  
- Identified inefficiencies in current operations such as static route planning and lack of real-time tracking.  
- Analyzed Salesforce features suitable for addressing the identified issues (CRM, Flow, Apex, Field Service Lightning).  
- Drafted the problem statement and proposed Salesforce-based solution approach.

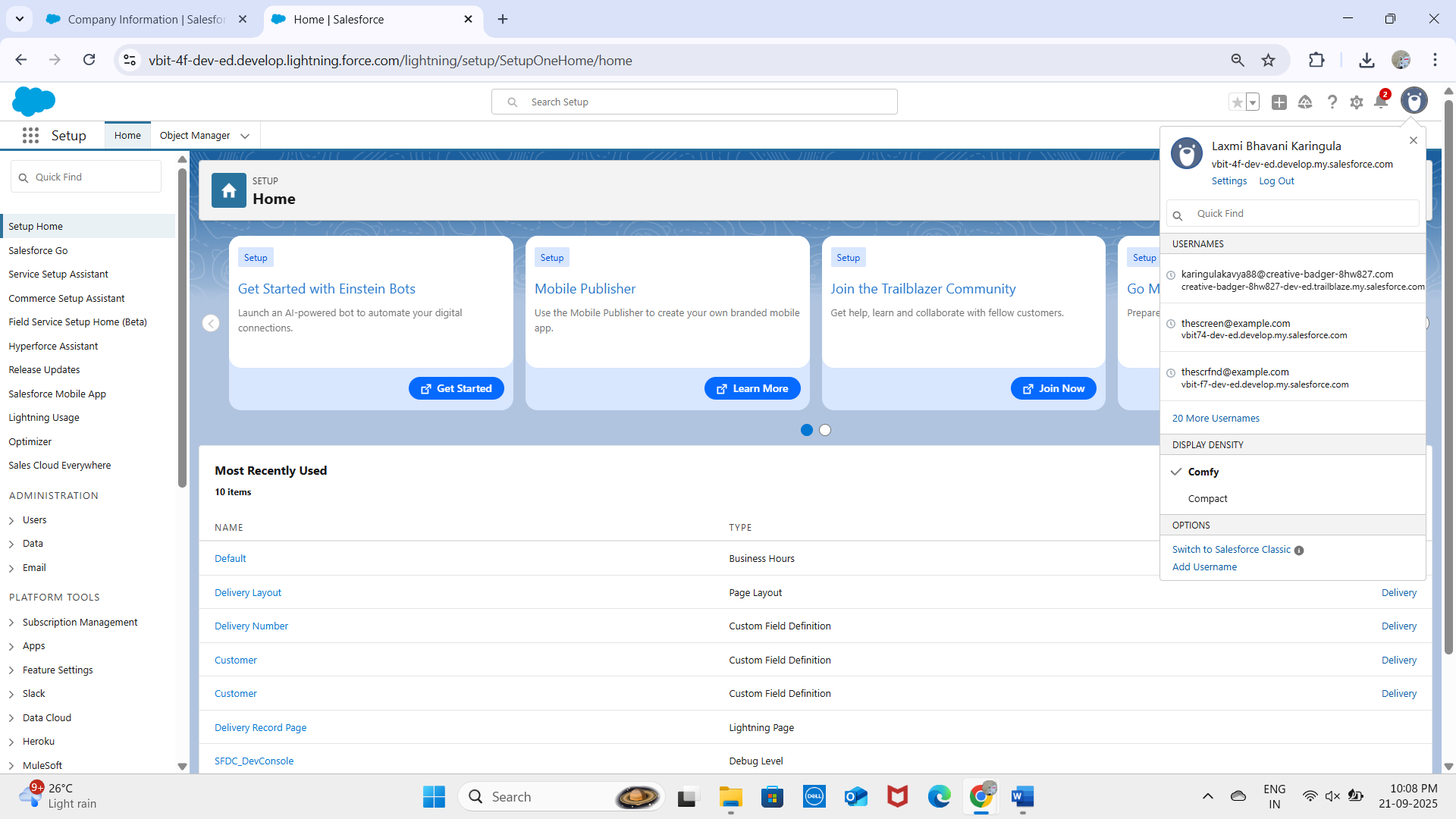
## Outcomes / Deliverables

- Clear understanding of logistics industry challenges in last-mile delivery.  
- Problem Statement document created.  
- High-level objectives of the Intelligent Last Mile Delivery Orchestration System defined.  
- Foundation laid for subsequent project phases.

# Phase 2: Salesforce Org Setup & Configuration

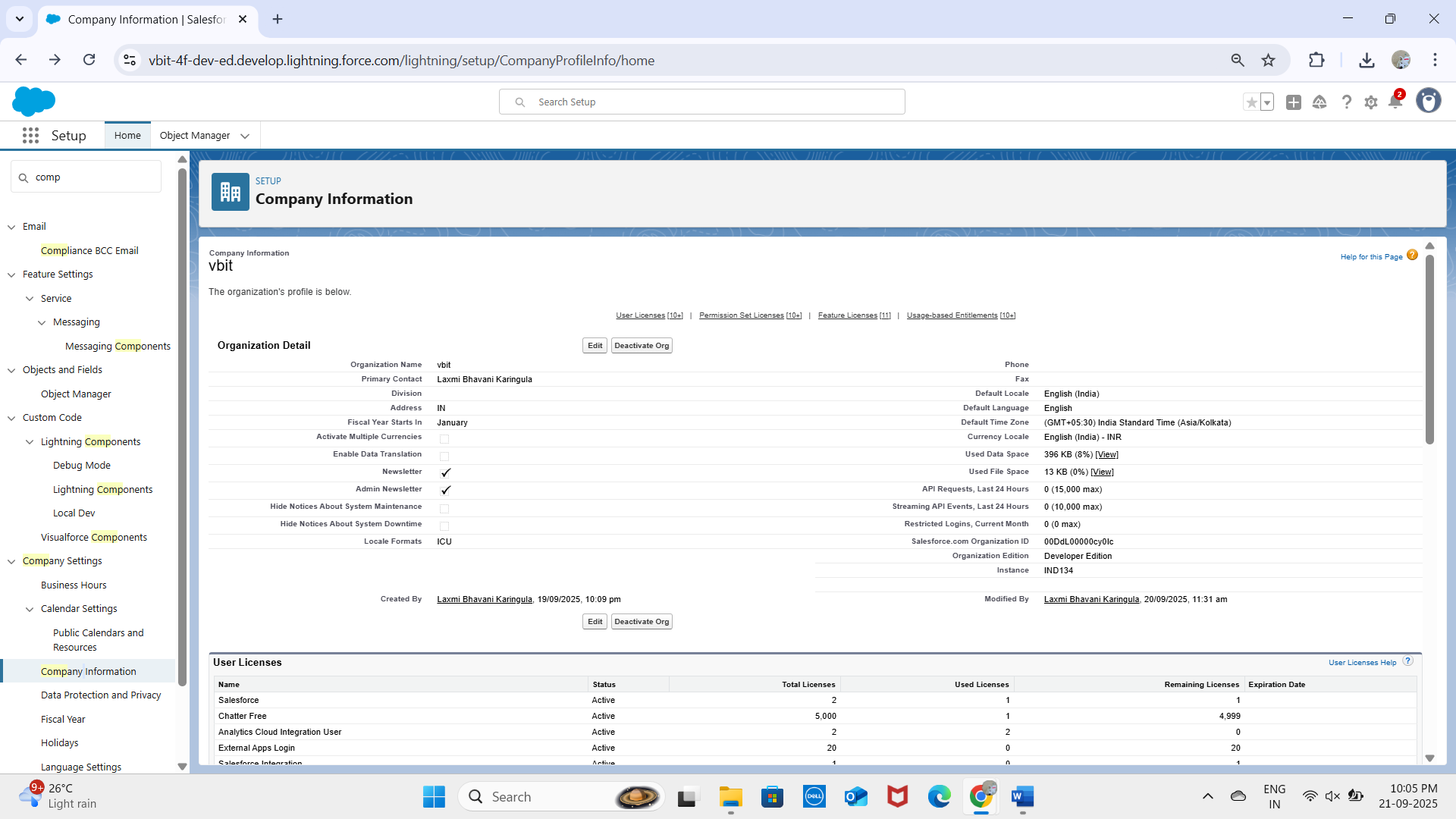
## 1. Salesforce Edition

We selected a Developer Edition Org for this project. It provides standard CRM features along with Apex, Lightning, Flows, APIs, and AppExchange access. The Developer Org was chosen because it is free, permanent, and supports all features required for delivery orchestration (objects, automation, integrations).



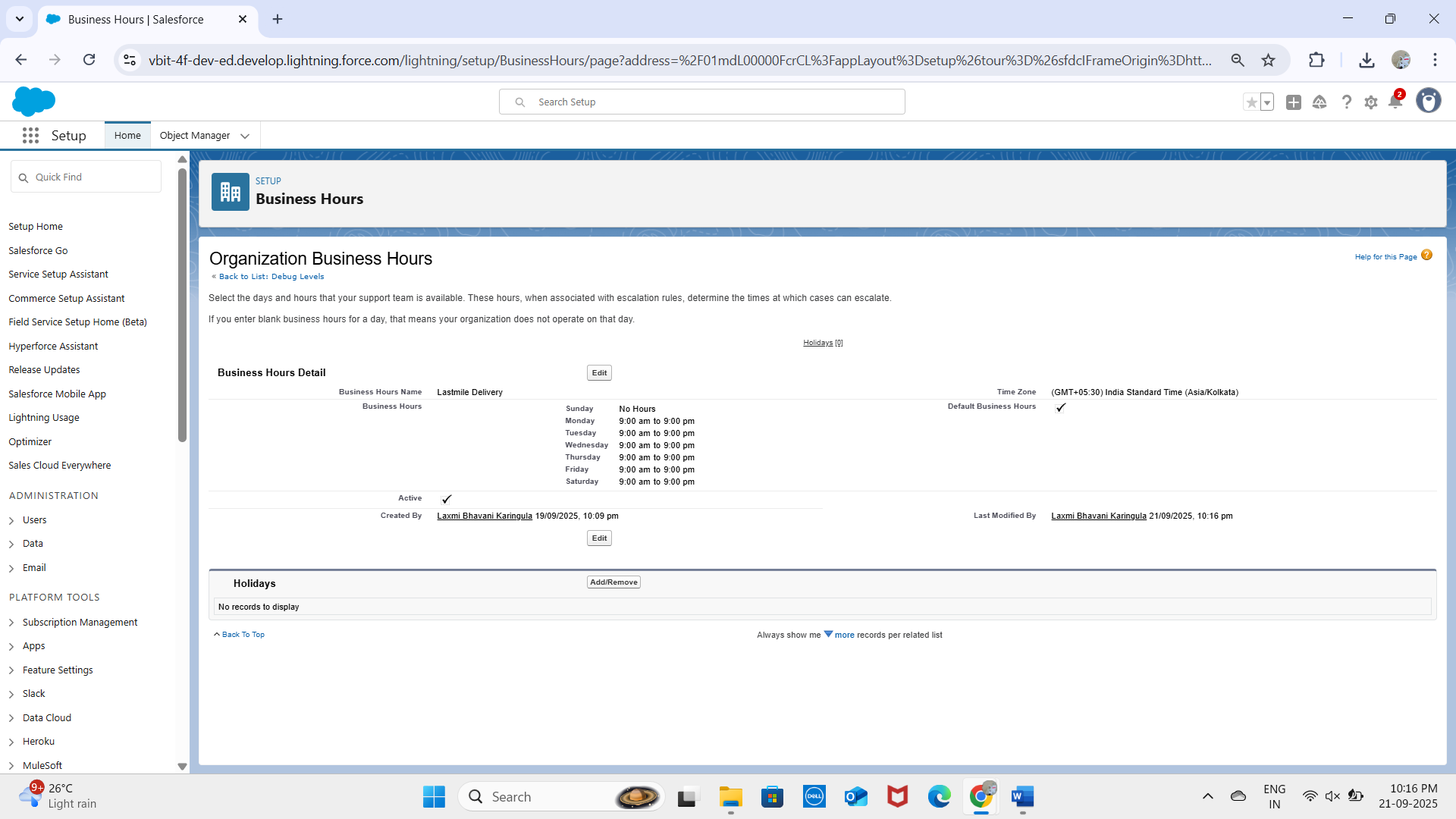
## 2. Company Profile Setup

Configured under Setup → Company Settings → Company Information.  
- Organization Name: vbit.  
- Time Zone: IST (India Standard Time).  
- Default Currency: INR.  
- Default Language: English.  
These settings ensure delivery schedules, SLA tracking, and reporting align with the region and business hours.



## 3. Business Hours & Holidays

Business hours defined as 9 AM – 9 PM, Monday to Saturday.  
Public holidays added to avoid scheduling deliveries on those dates.  
Ensures realistic planning for delivery agents and avoids assigning deliveries outside working hours.

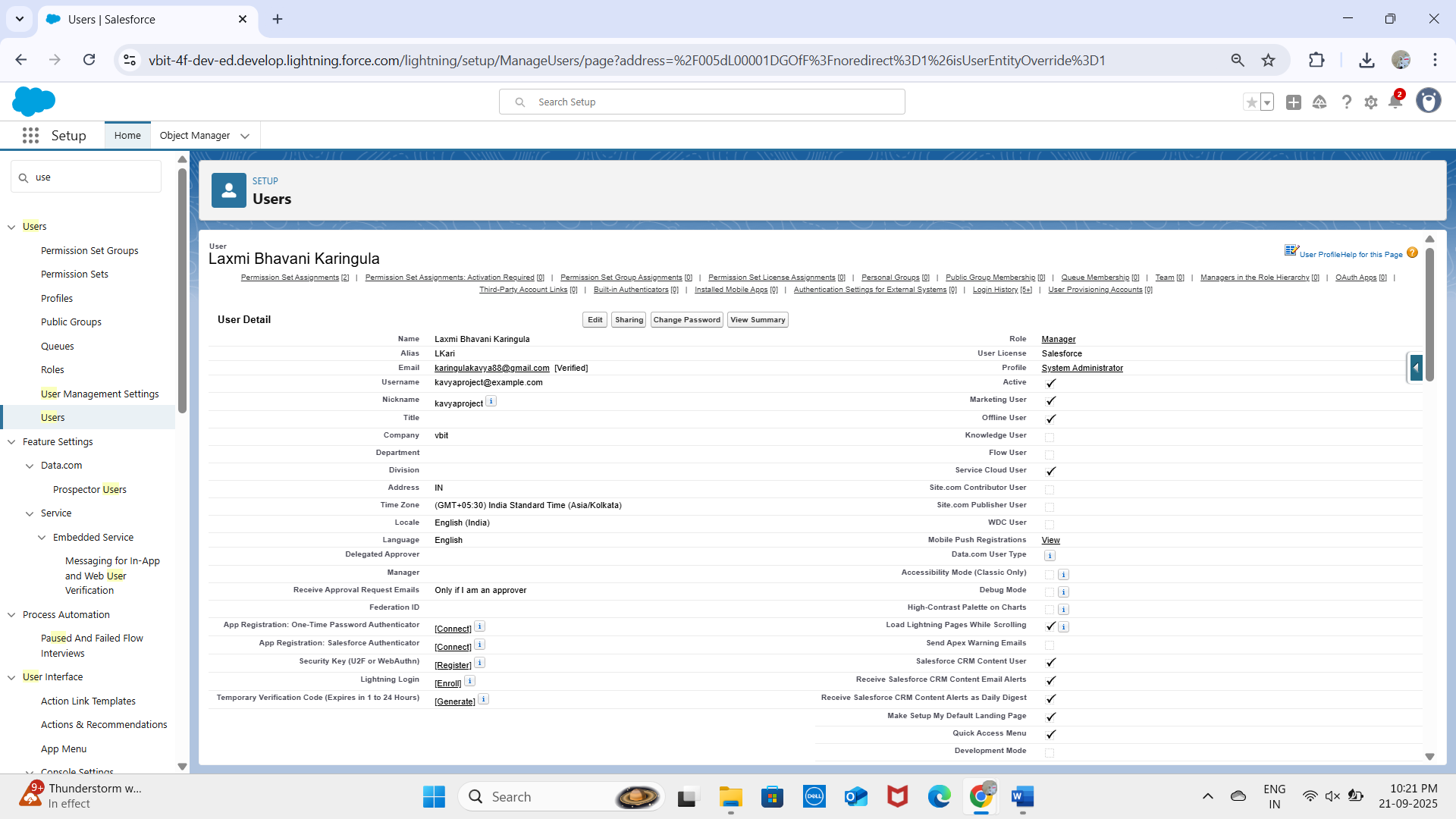


## 4. Fiscal Year Settings

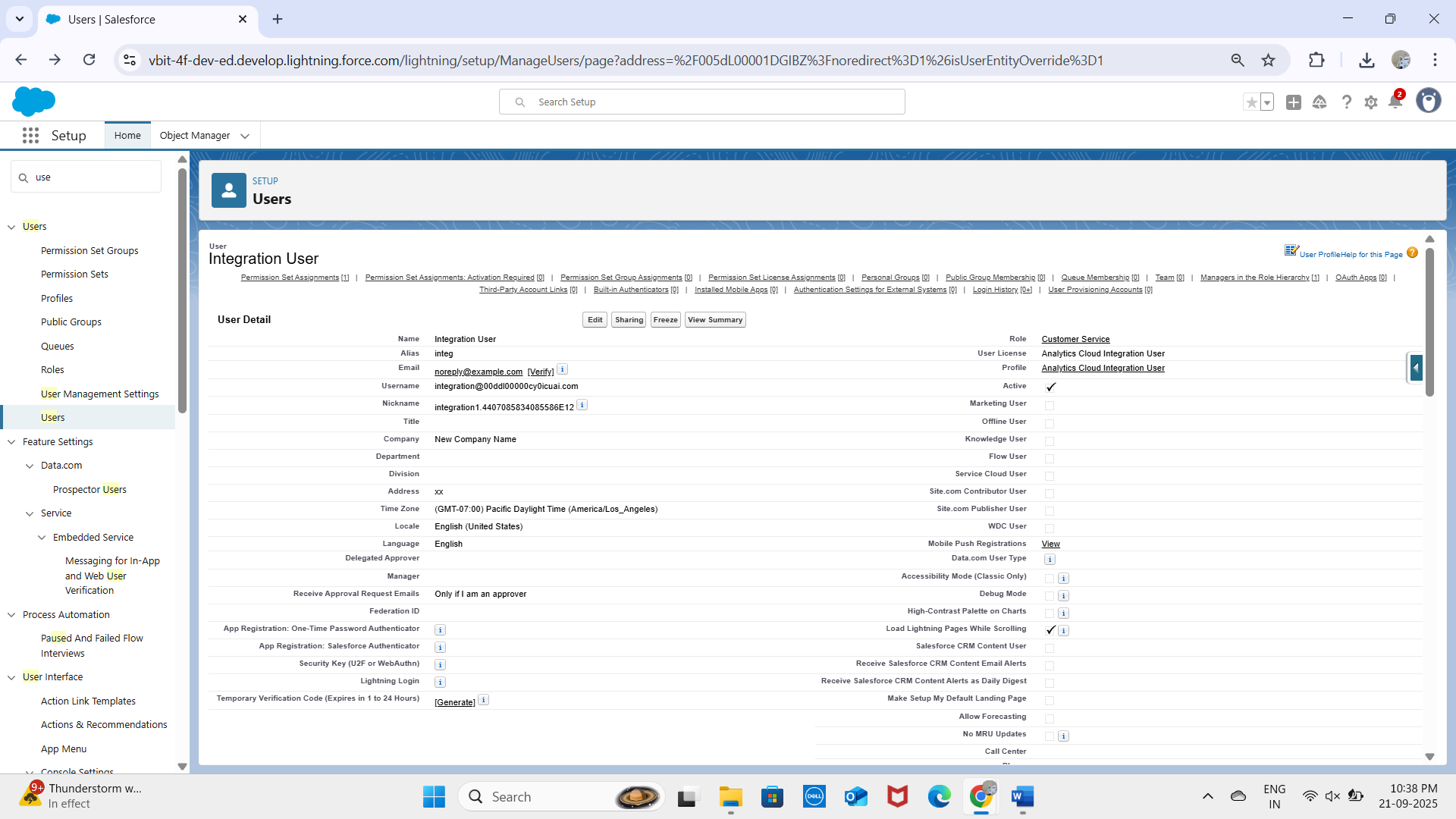
## We enabled the Standard Fiscal Year (January–December). This allows consistent tracking of delivery SLAs, agent performance, and customer satisfaction metrics across quarterly and yearly reports. No custom fiscal year was required for logistics reporting.

## 5. User Setup & Licenses

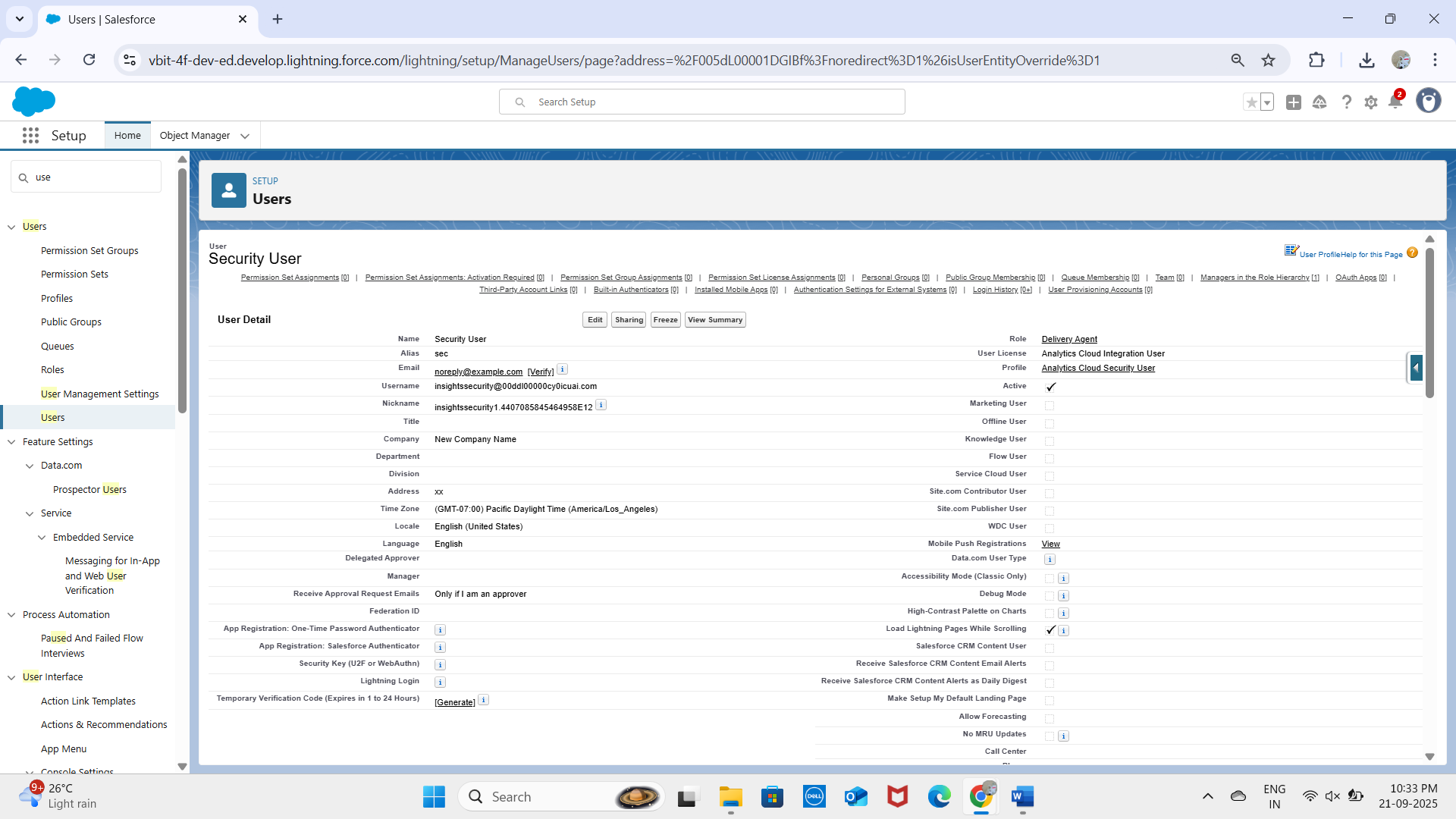
## Admin User created with System Administrator license.



* **Integration User**

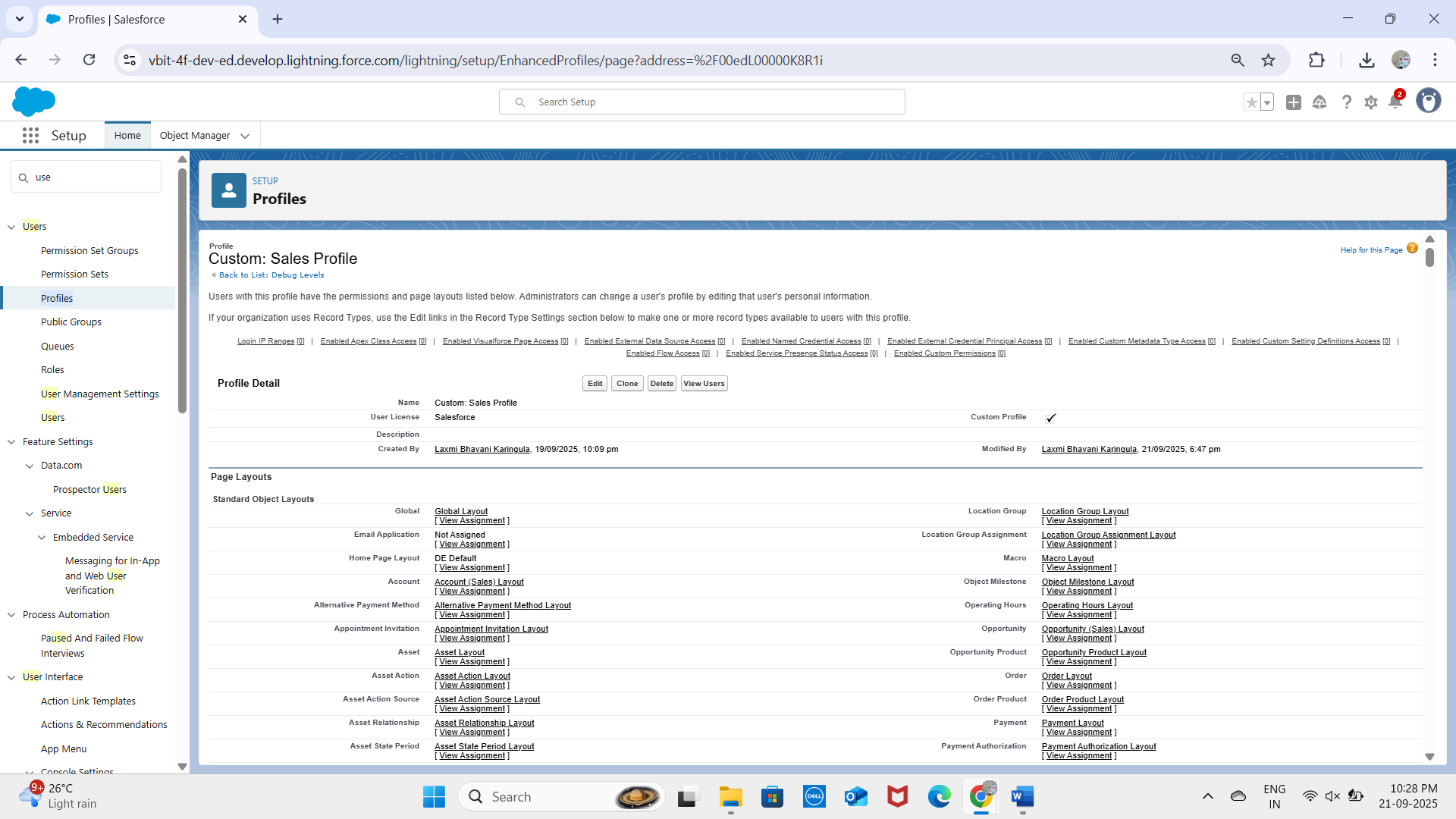


* **Security User**

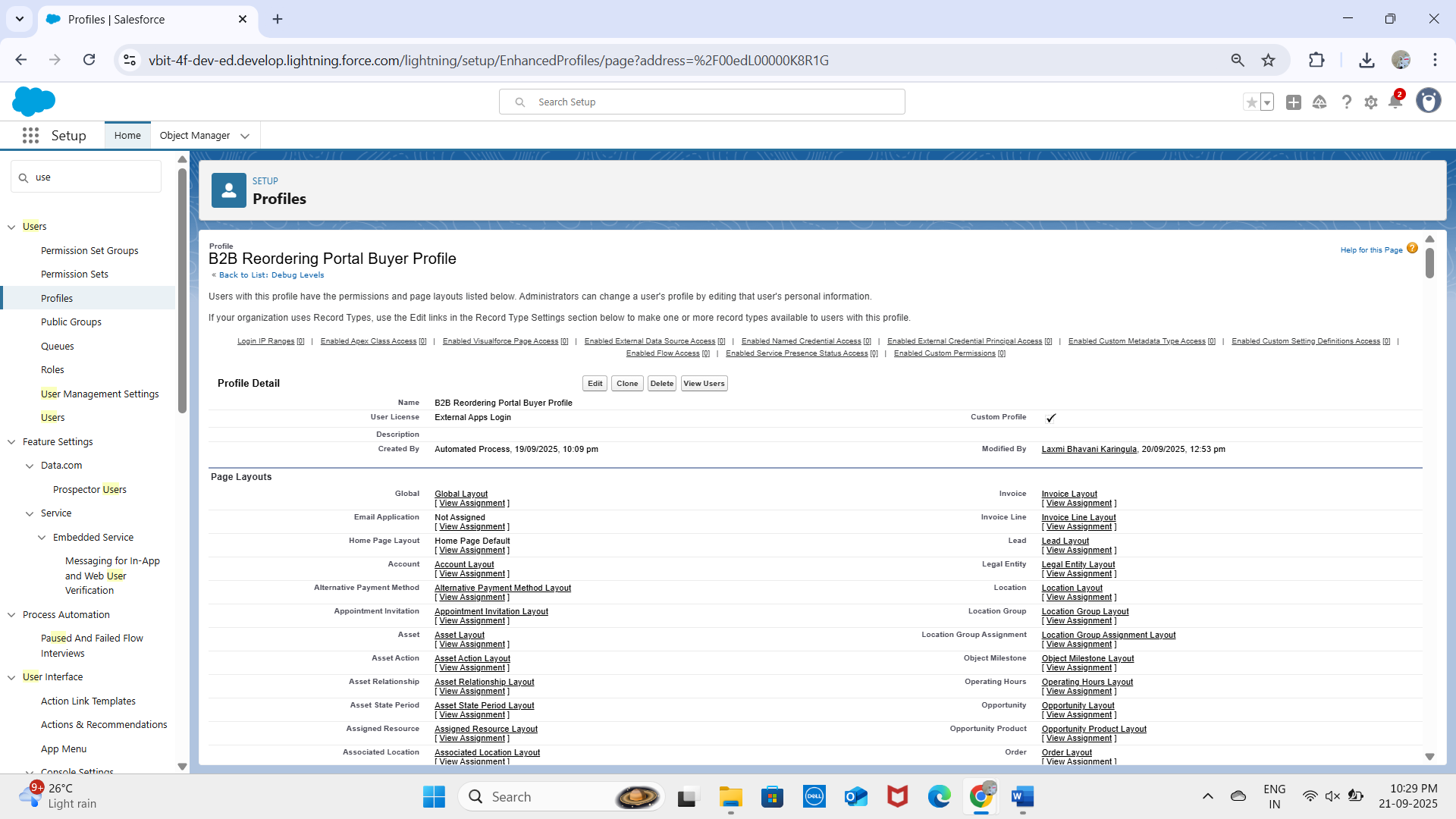
  
  
- All users verified via email, login tested successfully.

## 6. Profiles

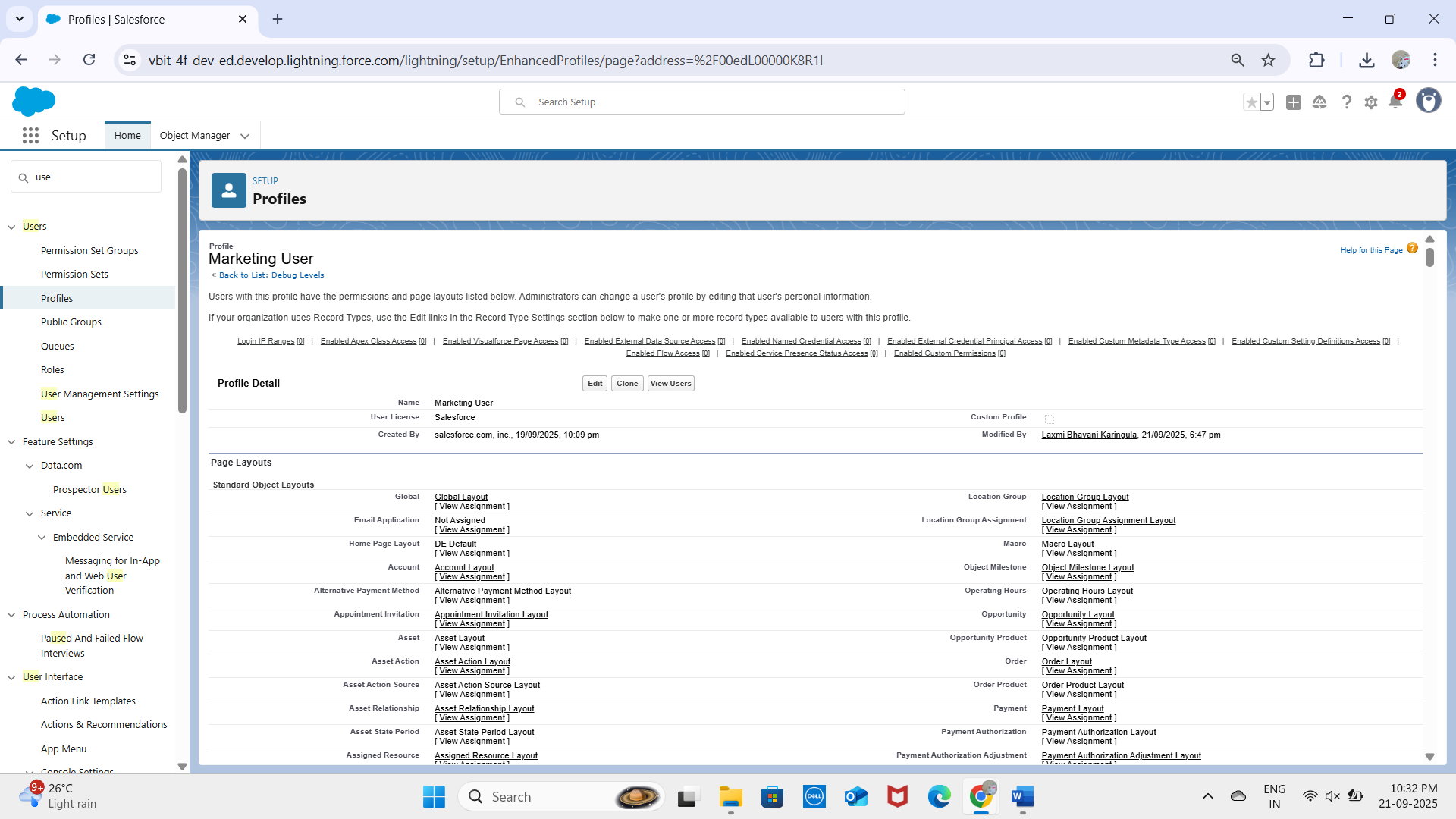
* **Admin Profile** – full access.
* **Manager Profile** – can oversee all deliveries, assign agents.
* **Customer Sales Profile** – can view and manage exceptions, notify customers.



* **Delivery Agent Profile** – restricted to update only their assigned deliveries and mark status.
* **B2B Reordering Portal Buyer Profile**

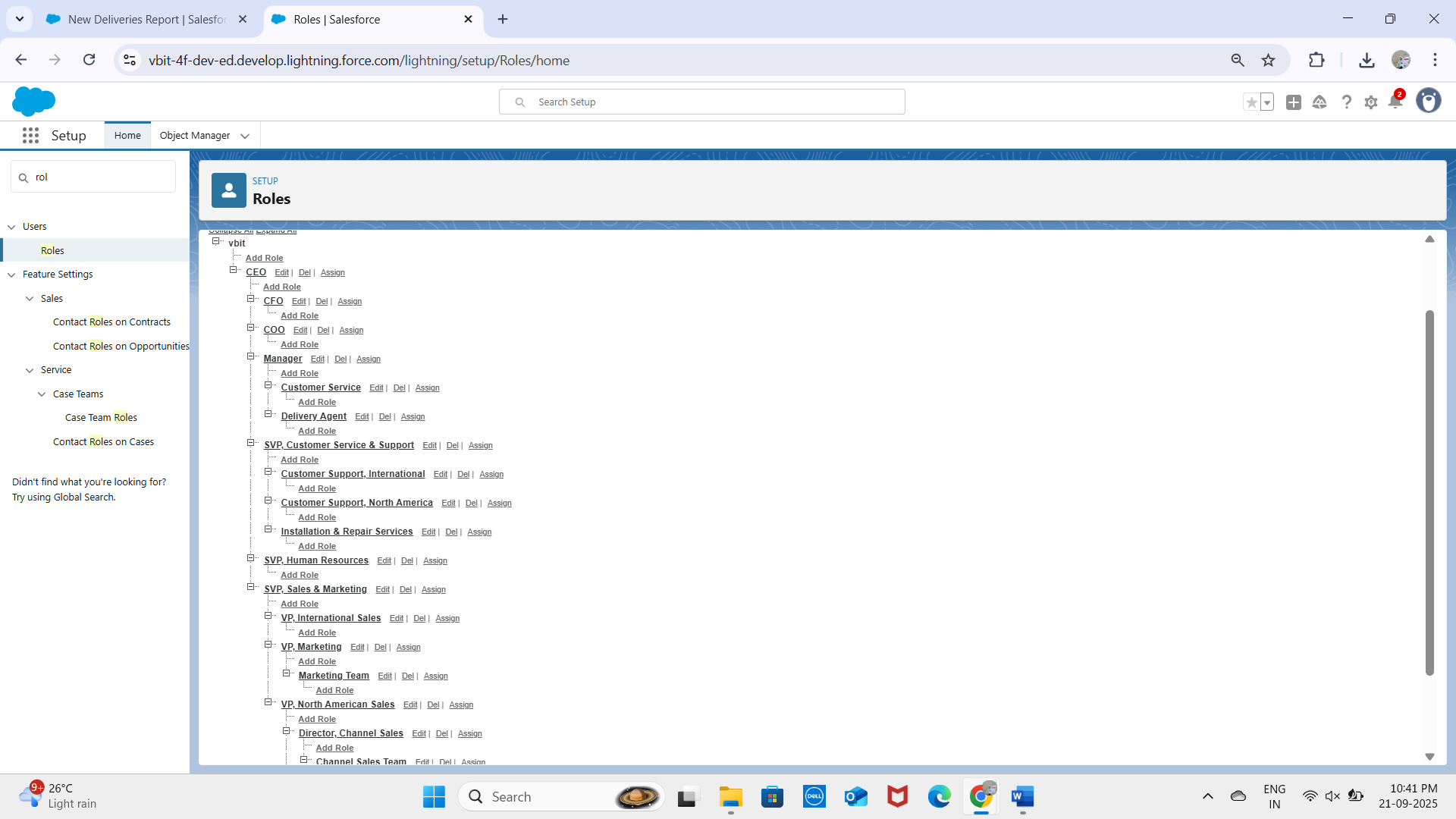


* **Marketing user**



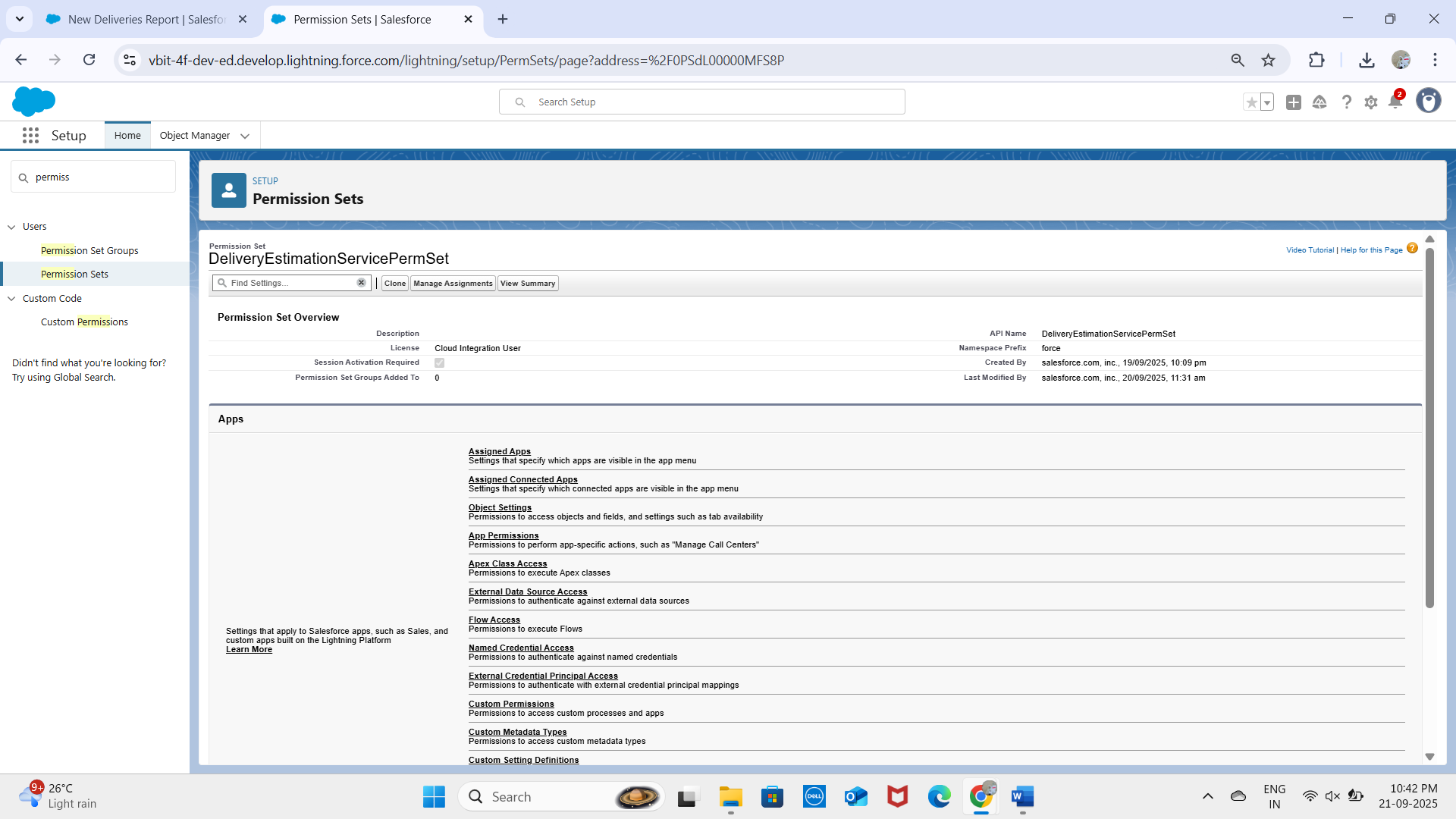
## 7. Roles

A role hierarchy was created:  
- Manager (top)  
 - Customer Service  
 - Delivery Agent  
This ensures visibility flows upwards: Managers see everything, CS can see relevant customer records, and Delivery Agents only see their own records.

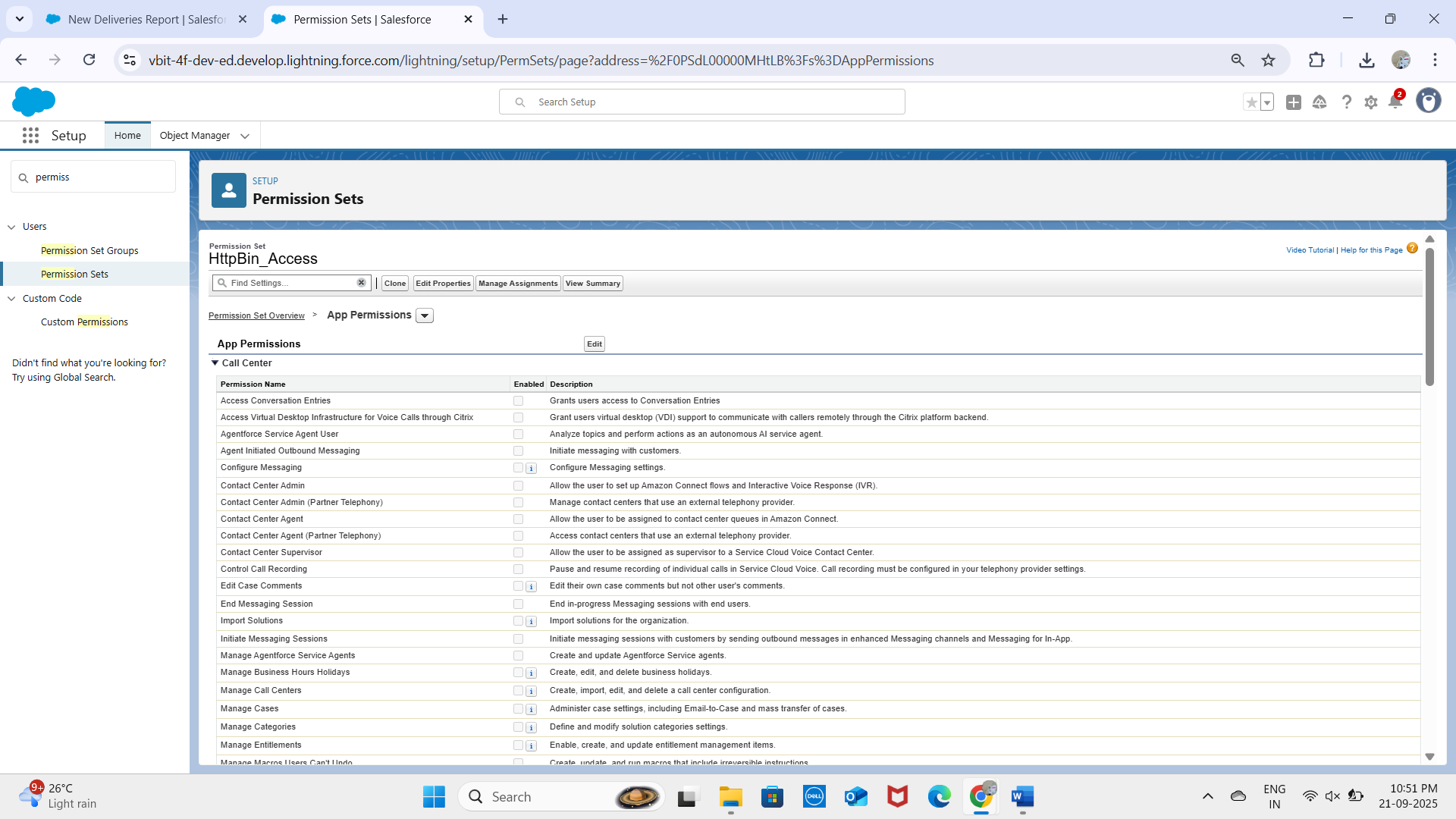


## 8. Permission Sets

Created a permission set 'DeliveryEstimationPermissionSet’to allow agents and CS staff to view dashboards without modifying profiles.  
Flexible extra access for testing integrations without changing role-based security.



Created HttpBin\_Access permission set

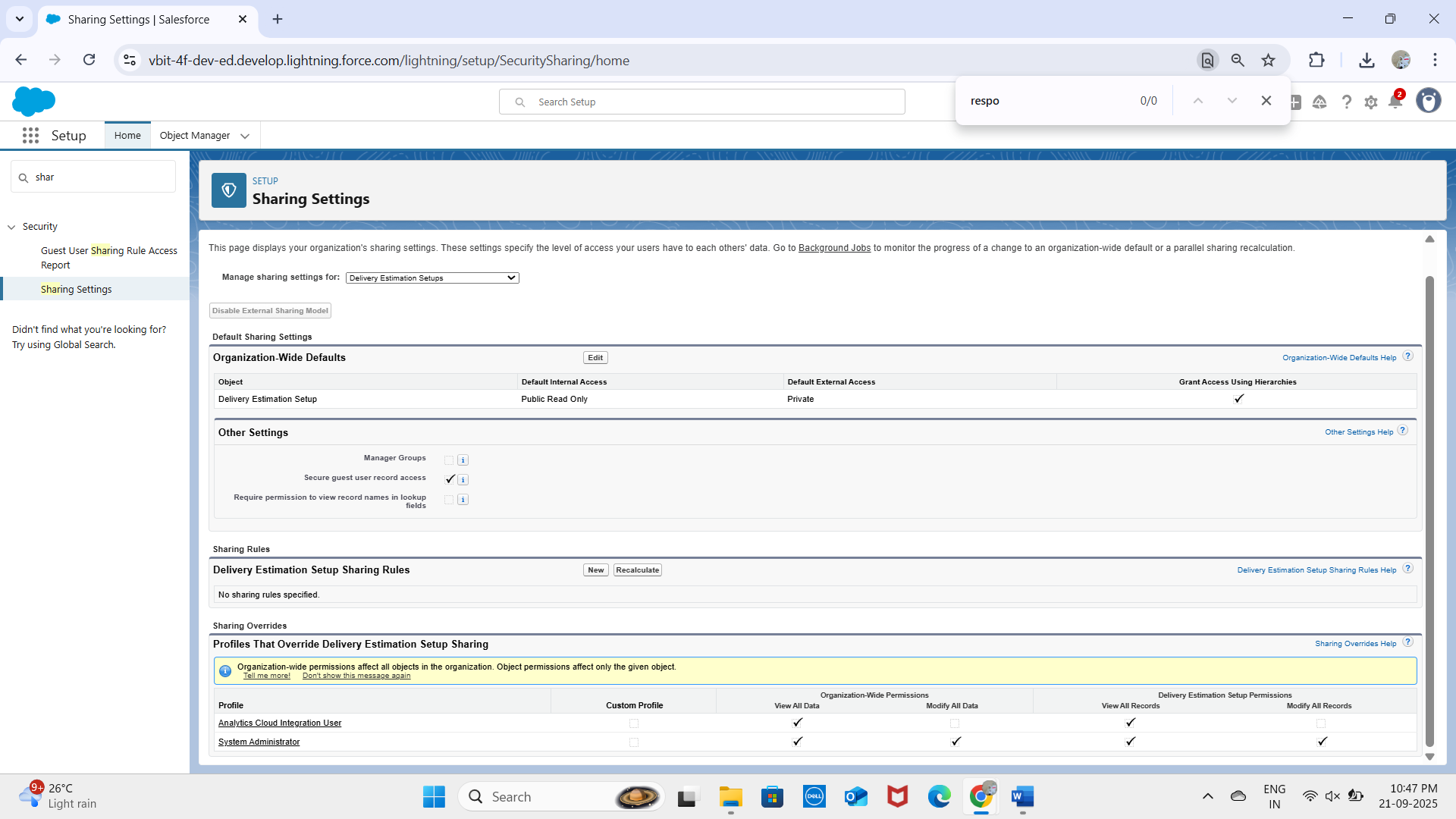


## 9. Organization-Wide Defaults (OWD)

- Delivery\_\_c – Private (only assigned agent + manager can access).  
- Route\_\_c – Public Read Only (visible for planning).  
- Delivery\_Exception\_\_c – Private (only agent and manager can view).  
- Notifications\_\_c – Public Read/Write (since system-generated alerts may need to be visible org-wide).

## 10. Sharing Rules

Delivery\_\_c records shared with Managers for oversight.  
Exceptions shared with Customer Service role for handling failed deliveries.  
This setup ensures operational transparency while maintaining data security.

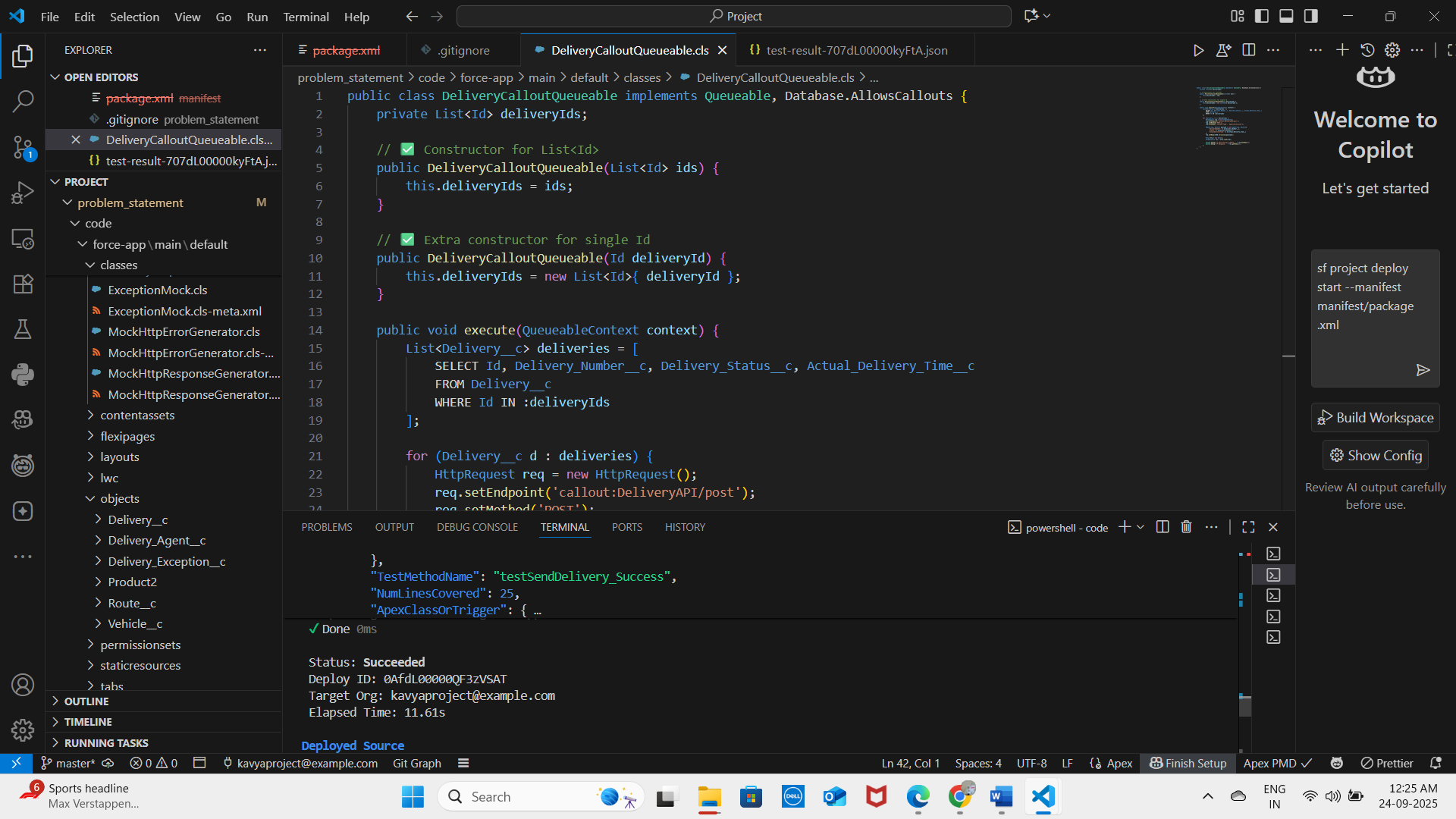


## 11. Login Access Policies

- Delivery Agents: login restricted to 8 AM – 10 PM.  
- Managers & CS: unrestricted login.  
- Two-Factor Authentication (2FA) enabled for all profiles.  
Ensures system security while allowing flexible access.

## 12. Deployment Basics

Custom objects created and deployed:  
- Delivery\_\_c – core object for tracking packages.  
- Route\_\_c – stores route and address details.  
- Delivery\_Exception\_\_c – logs exceptions (delays, failures).  
- Notifications\_\_c – tracks customer notifications.

  
Deployment options prepared using Change Sets and SFDX (VS Code).  
Both were tested to confirm successful migration of metadata.

✅ This completed the Salesforce Org Setup & Configuration for the Last-Mile Delivery Orchestration system.

**Phase 3: Data Modeling & Relationships**

* **Standard & Custom Objects**

The data model for the Intelligent Last Mile Delivery Orchestration System combines standard Salesforce objects with newly created custom objects to support delivery operations, routing, and customer communication.

**Standard Objects:**

- Account: Represents the business customers or clients associated with deliveries.  
- Contact: Represents individual customers receiving deliveries.  
- User: Represents delivery agents, managers, and customer service staff.

**Custom Objects:**

- Delivery\_\_c: Tracks each delivery record with status, scheduling, and completion details.  
- Route\_\_c: Holds optimized route details for delivery assignments.  
- Delivery\_Exception\_\_c: Captures failed or delayed deliveries and rescheduling information.  
- Notifications\_\_c: Stores automated communication alerts sent to customers.

* **Fields & Relationships**

Custom fields capture the logistics data required to manage deliveries, while relationships link the objects together.

**Relationships:**

- Delivery\_\_c → Lookup to Account (Customer placing the delivery).  
- Delivery\_\_c → Lookup to User (Assigned Delivery Agent).  
- Route\_\_c → Master-Detail with Delivery\_\_c (one route may cover multiple deliveries).  
- Delivery\_Exception\_\_c → Lookup to Delivery\_\_c (each exception tied to a specific delivery).  
- Notifications\_\_c → Lookup to Delivery\_\_c (notifications linked to delivery record).

**Custom Fields:**

On Delivery\_\_c:  
- Delivery\_Number\_\_c (Auto Number)  
- Delivery\_Status\_\_c (Picklist: Scheduled, In Transit, Delivered, Failed)  
- Scheduled\_Time\_\_c (Date/Time)  
- Actual\_Delivery\_Time\_\_c (Date/Time)  
- Customer\_\_c (Lookup to Account)  
- Assigned\_Agent\_\_c (Lookup to User)  
  
On Route\_\_c:  
- Route\_Name\_\_c (Text)  
- Optimized\_Path\_\_c (Long Text)  
- Total\_Distance\_\_c (Number)  
- Estimated\_Time\_\_c (Number)  
  
On Delivery\_Exception\_\_c:  
- Exception\_Type\_\_c (Picklist: Delay, Customer Not Available, Traffic, Weather)  
- Exception\_Details\_\_c (Text Area)  
- Rescheduled\_Time\_\_c (Date/Time)  
  
On Notifications\_\_c:  
- Notification\_Type\_\_c (Picklist: SMS, Email, In-App)  
- Message\_Content\_\_c (Text Area)  
- Sent\_Time\_\_c (Date/Time)

* **User Interface & Layouts**

Page Layouts: Custom layouts were created for Delivery, Route, Exception, and Notifications to display relevant fields and related lists clearly for different roles (Agents, Managers, Service Staff).  
  
Compact Layouts: Configured for Delivery\_\_c to show Delivery Number, Status, and Scheduled Time in highlights panel. Other compact layouts created for Route\_\_c and Delivery\_Exception\_\_c.  
  
Record Types: Not required in the initial phase but can be added later to distinguish different delivery types (e.g., Standard Delivery, Express Delivery).

* **Data Architecture**

Schema Builder: Used to visualize all objects and their relationships, showing Account and User at the center with custom objects linked for deliveries, routes, and notifications.  
  
Junction Objects: Not required for the current design. However, if a Delivery\_\_c could belong to multiple routes, a junction object such as Delivery\_Route\_Assignment\_\_c would be created with master-detail relationships to both Delivery\_\_c and Route\_\_c.

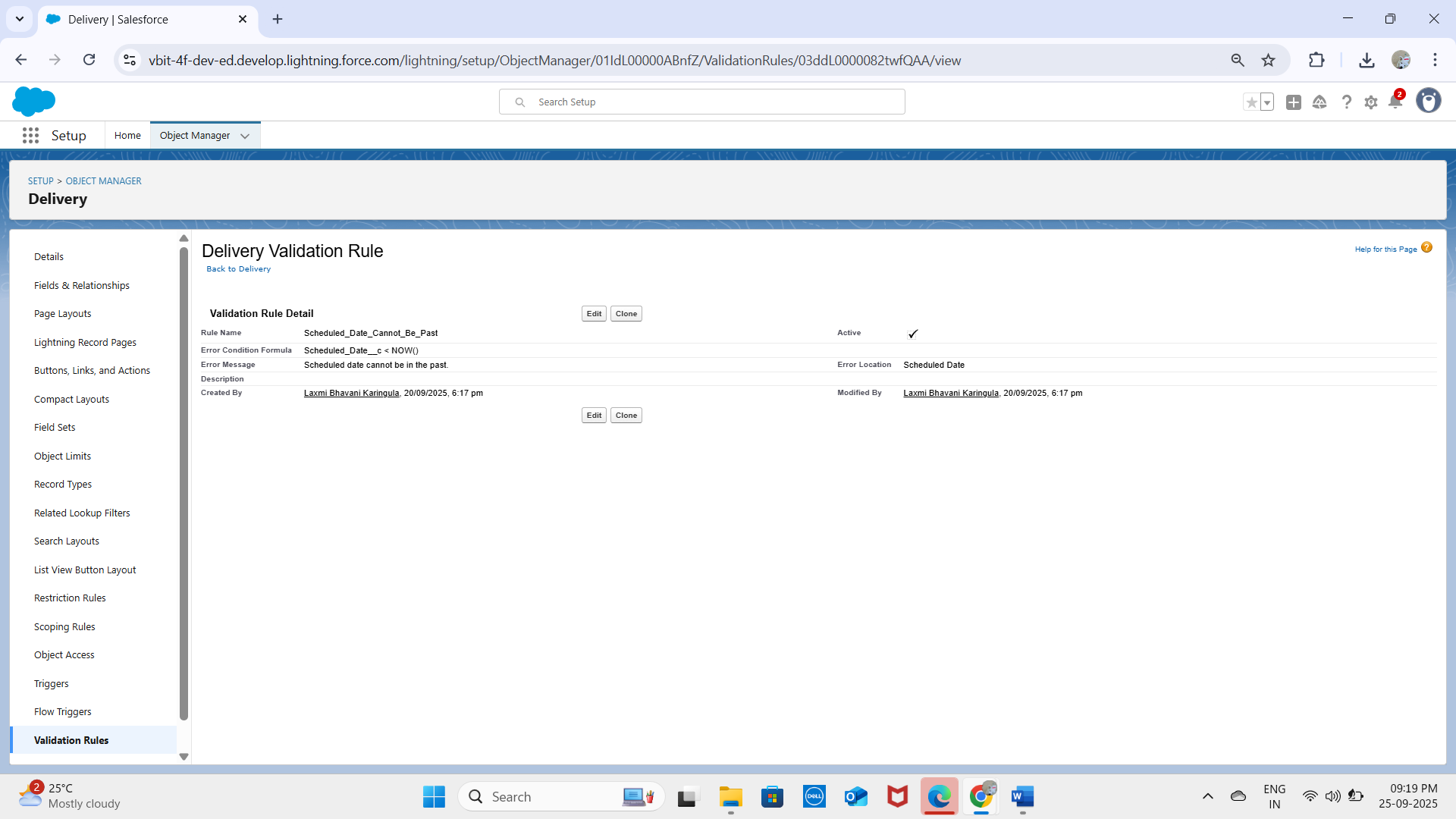
# PHASE 4 – Process Automation (Admin)

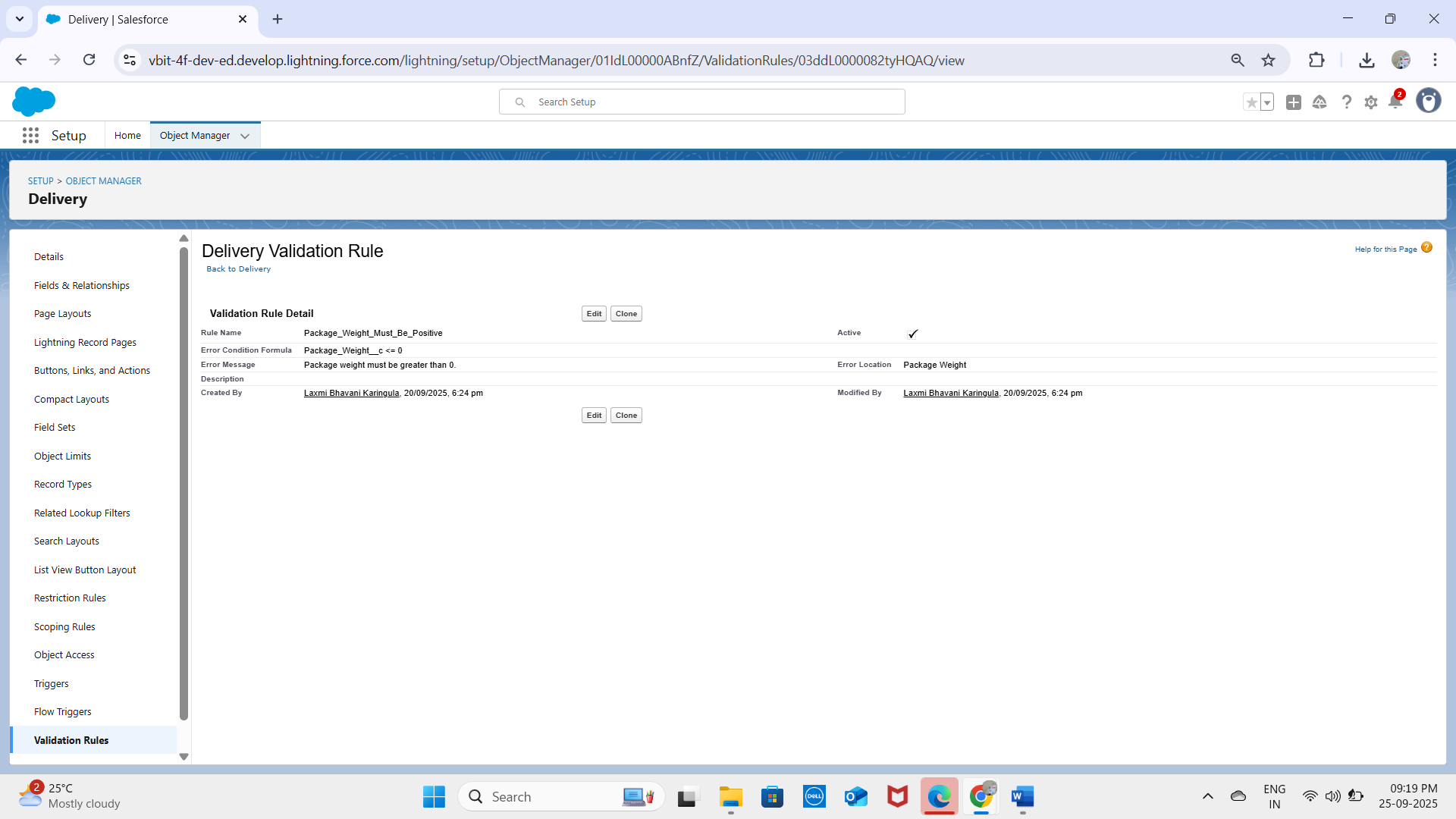
The goal of this phase is to use Salesforce's declarative tools to automate delivery operations, enforce data quality, and handle business logic using Flows and Validation Rules.

## Validation Rules

Validation rules were created to ensure clean and accurate data entry.

- Object: Delivery\_\_c  
- Rule: Scheduled date cannot be in the past.  
- Formula:  
 Scheduled\_Date\_\_c < NOW()  
- Error Message: "Scheduled date cannot be in the past."





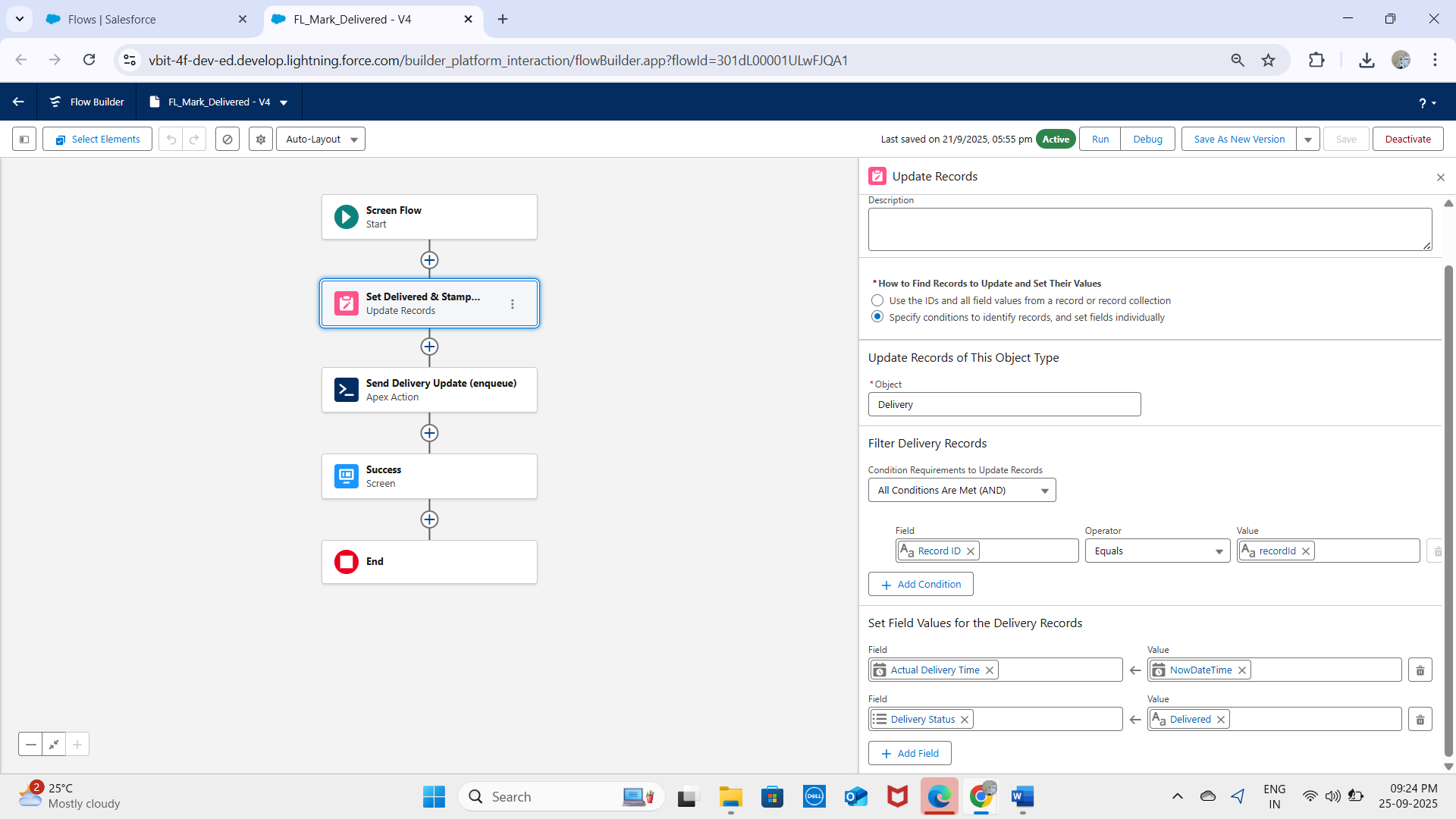
- Object: Order\_\_c  
- Rule: Ensure Total\_Amount\_\_c is greater than zero.  
- Formula:  
 Total\_Amount\_\_c <= 0  
- Error Message: "Total Amount must be greater than zero."

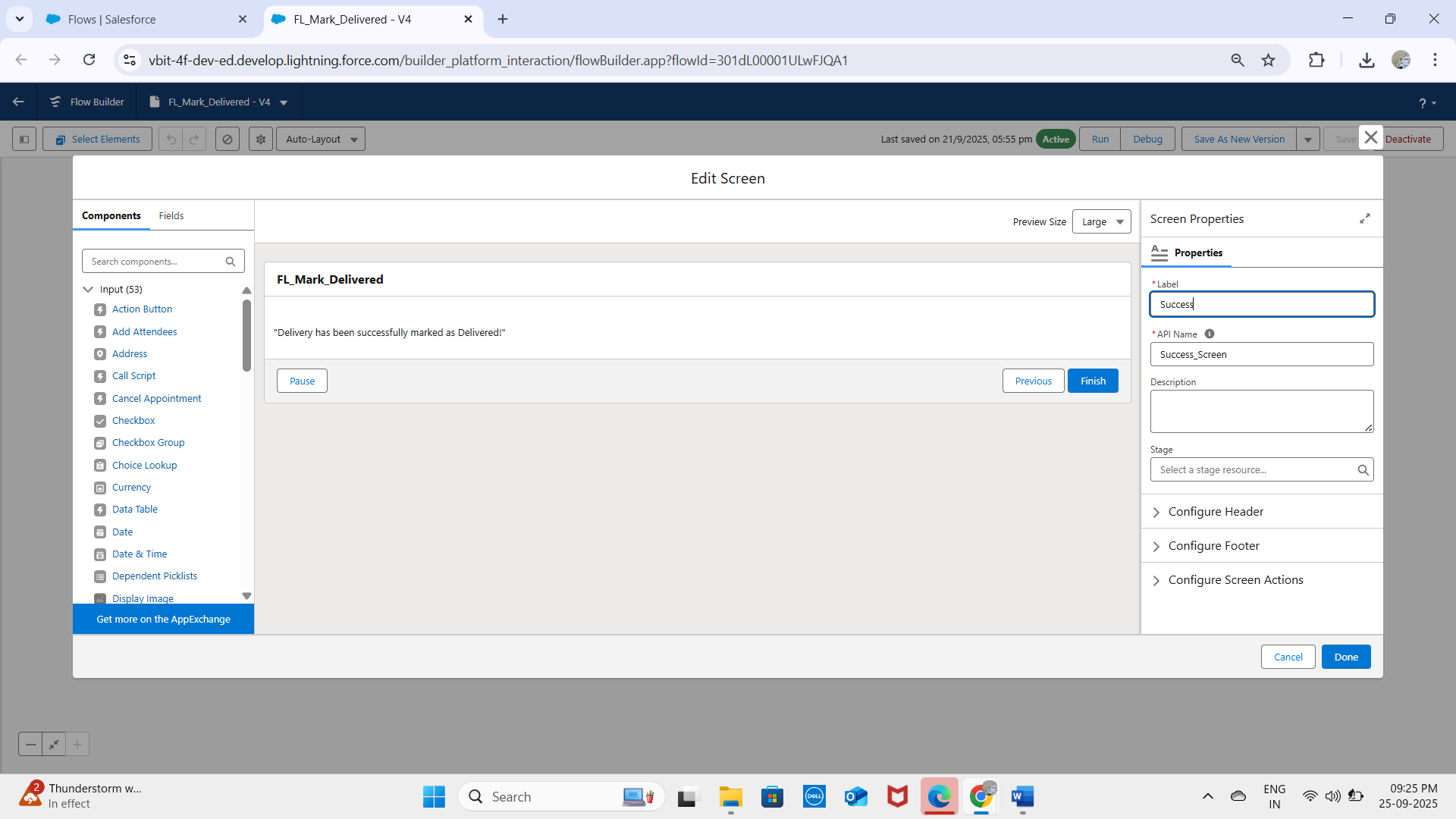
## Flow Builder (Record-Triggered Flow)

Flows were used as the primary automation tool for project logic.

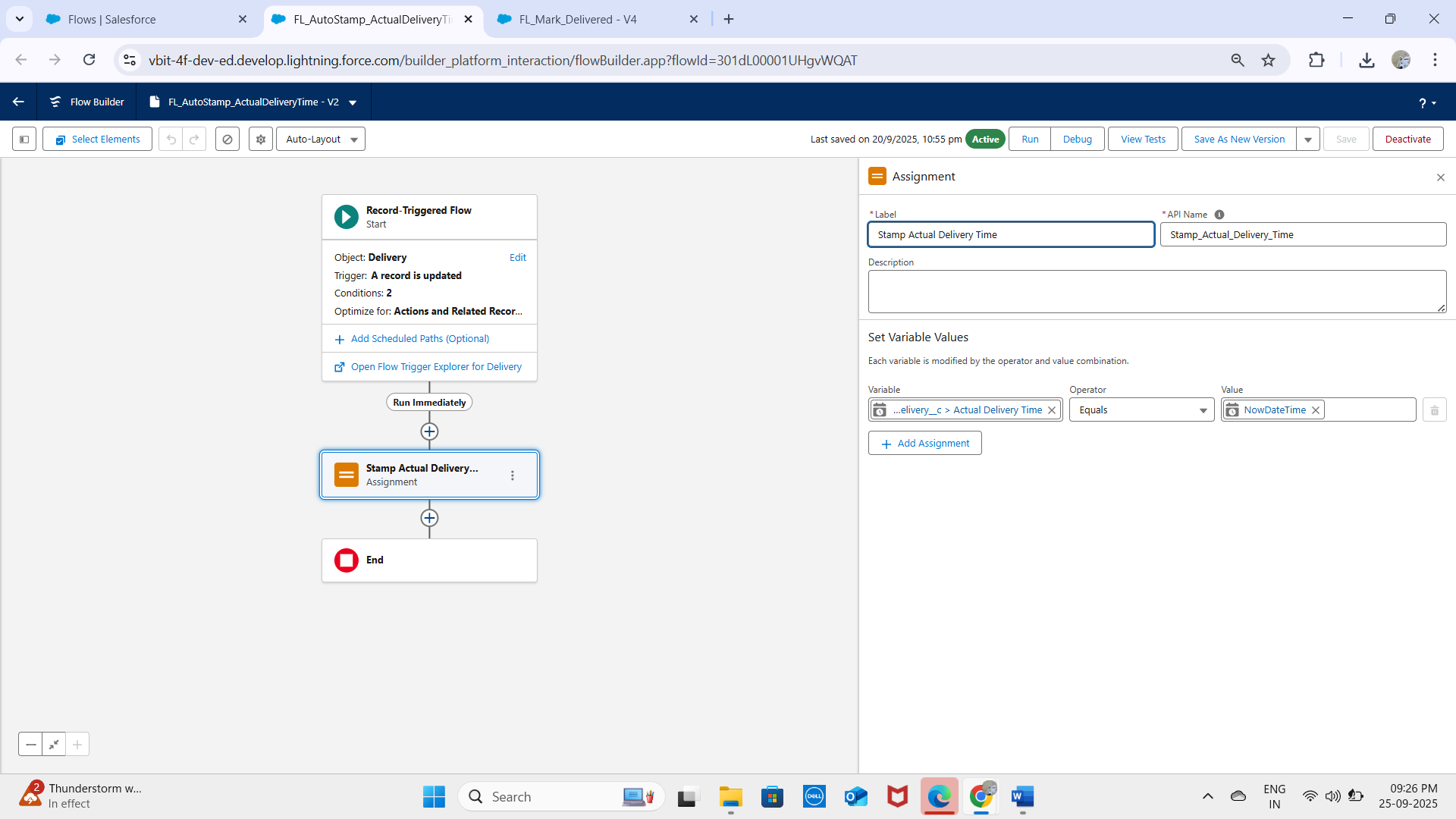
### Flow 1 – Update Delivery Status

- Objective: Automatically update delivery status when order is marked complete.  
- Trigger: Record-Triggered Flow on Order\_\_c → when Status\_\_c = 'Complete'.  
- Logic:  
 1. Get Records → Find related Delivery\_\_c record using Order\_\_c.  
 2. Update Records → Set Delivery\_Status\_\_c = 'Delivered'.

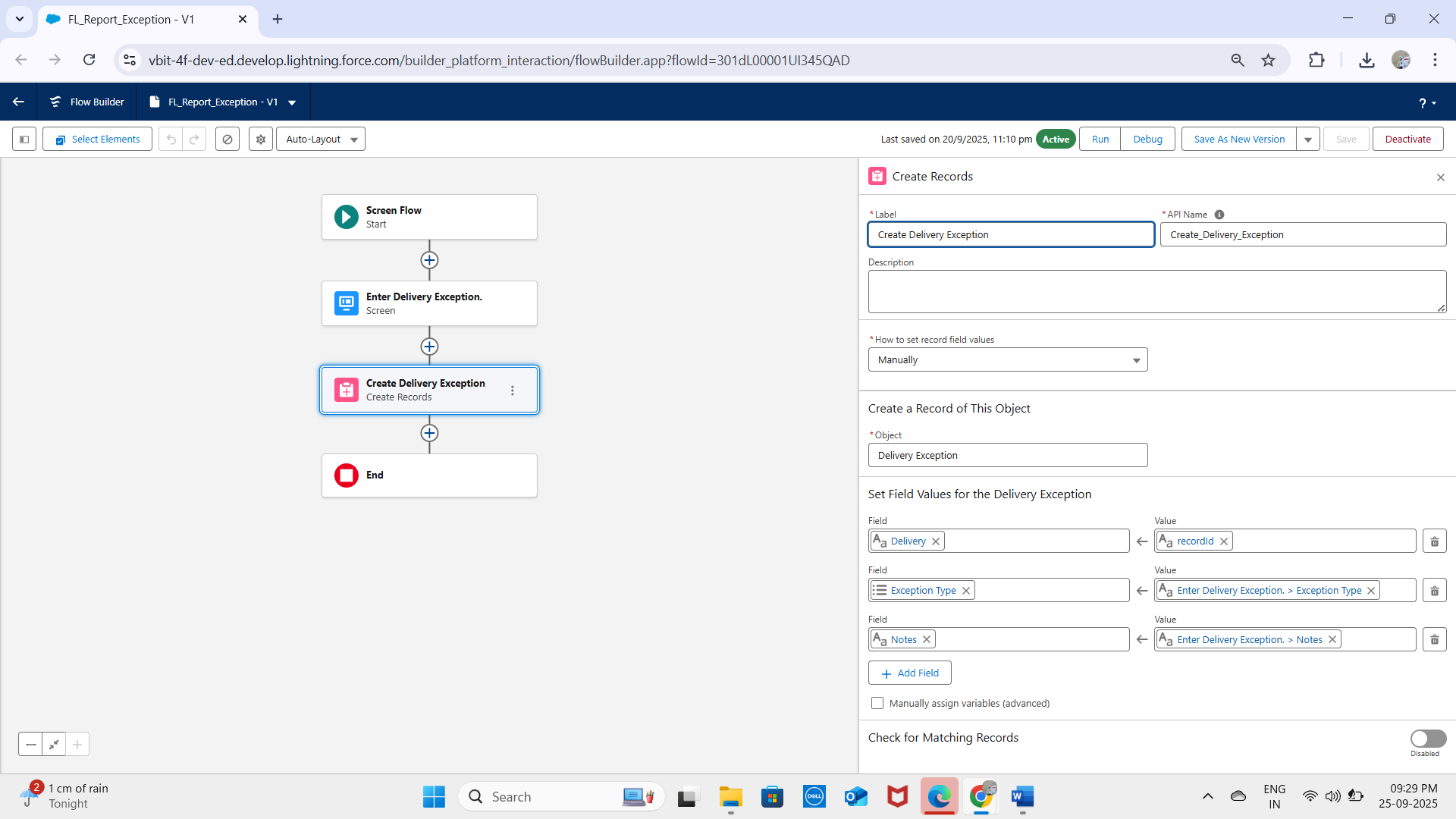




### Flow 2 – AutoStamp ActualDelivery



**Flow 3 -Report Exception**



## Approval Process

A future requirement was identified to ensure manager approvals for high-value orders.  
  
- Object: Order\_\_c  
- Criteria: If Total\_Amount\_\_c > 10,000 INR  
- Process:  
 • Submit automatically to Manager for Approval.  
 • Manager approves/rejects.  
 • Status updates accordingly.

## Workflow Rules & Process Builder

- No new workflows or Process Builder were created (deprecated tools).  
- All automation for the project (Deliveries, Orders, Loyalty) was implemented with Flows following Salesforce best practices.

## Outcome

With this phase, the project now has:  
- Data quality enforcement (Validation Rules)  
- Business logic automation (Flows)  
- Governance workflow (Approval Process)

# PHASE 5 – Apex Programming (Developer)

In this phase, we extend beyond Salesforce’s point-and-click tools and use Apex programming to implement custom business logic for Deliveries, Orders, and Loyalty Points in the CRM. Apex ensures scalability, efficiency, and complex automation where declarative tools are insufficient.

## 1. Apex Classes & Objects

Apex classes were created to handle reusable logic.  
  
Example – Confirm Delivery

**public class DeliveryService {  
 public static void confirmDelivery(Id deliveryId) {  
 Delivery\_\_c d = [SELECT Id, Delivery\_Status\_\_c FROM Delivery\_\_c WHERE Id = :deliveryId];  
 d.Delivery\_Status\_\_c = 'Confirmed';  
 update d;  
 }  
}**

## 2. Apex Triggers

Triggers automate processes when records are inserted, updated, or deleted.  
Example – Prevent Duplicate Delivery Numbers

trigger DeliveryTrigger on Delivery\_\_c (before insert, before update) {  
 for (Delivery\_\_c d : Trigger.new) {  
 Boolean exists = [  
 SELECT Id FROM Delivery\_\_c   
 WHERE Delivery\_Number\_\_c = :d.Delivery\_Number\_\_c   
 AND Id != :d.Id  
 LIMIT 1  
 ] != null;  
 if (exists) {  
 d.addError('This Delivery Number already exists.');  
 }  
 }  
}

## 3. Trigger Design Pattern

Best practice is to move logic into a handler class to keep triggers clean.

trigger DeliveryTrigger on Delivery\_\_c (before insert, before update) {  
 DeliveryHandler.beforeSave(Trigger.new, Trigger.oldMap);  
}  
  
public class DeliveryHandler {  
 public static void beforeSave(List<Delivery\_\_c> newList, Map<Id, Delivery\_\_c> oldMap) {  
 // custom validation logic here  
 }  
}

## 4. SOQL & SOSL

SOQL – Query Deliveries by Status

List<Delivery\_\_c> deliveries = [SELECT Id, Delivery\_Status\_\_c FROM Delivery\_\_c WHERE Delivery\_Status\_\_c = 'Pending'];  
SOSL – Search Across Customers and OrdersList<List<SObject>> results =   
 [FIND 'Kavya\*' IN ALL FIELDS   
 RETURNING Customer\_\_c(Id, Name), Order\_\_c(Id, Status\_\_c)];

## 5. Collections (List, Set, Map)

List<String> orderIds = new List<String>();  
Set<Id> customerIds = new Set<Id>();  
Map<Id, Order\_\_c> orderMap = new Map<Id, Order\_\_c>([SELECT Id, Status\_\_c FROM Order\_\_c]);

## 6. Control Statements

for (Order\_\_c o : [SELECT Id, Status\_\_c FROM Order\_\_c]) {  
 if (o.Status\_\_c == 'Pending') {  
 o.Status\_\_c = 'Completed';  
 }  
}

## 7. Batch Apex

For large data processing (e.g., updating multiple deliveries).

global class UpdateDeliveryBatch implements Database.Batchable<SObject> {  
 global Database.QueryLocator start(Database.BatchableContext bc) {  
 return Database.getQueryLocator('SELECT Id, Delivery\_Status\_\_c FROM Delivery\_\_c');  
 }  
 global void execute(Database.BatchableContext bc, List<Delivery\_\_c> scope) {  
 for (Delivery\_\_c d : scope) {  
 d.Delivery\_Status\_\_c = 'Archived';  
 }  
 update scope;  
 }  
 global void finish(Database.BatchableContext bc) {}  
}

## 8. Queueable Apex

public class NotifyCustomerJob implements Queueable {  
 public void execute(QueueableContext context) {  
 // send email/notification about delivery update  
 }  
}

## 9. Scheduled Apex

global class DailyDeliveryReminder implements Schedulable {  
 global void execute(SchedulableContext sc) {  
 // send daily reminders for pending deliveries  
 }  
}

## 10. Future Methods

public class PaymentGateway {  
 @future(callout=true)  
 public static void processPayment(Id paymentId) {  
 // external API call for payment processing  
 }  
}

## 11. Exception Handling

try {  
 // risky logic  
} catch (Exception e) {  
 System.debug('Error: ' + e.getMessage());  
}

## 12. Test Classes

All Apex logic must be covered by test classes before deployment.

@isTest  
private class DeliveryTriggerTest {  
 @isTest static void testDuplicateDeliveryNumber() {  
 // insert deliveries and check addError  
 }  
}

## 13. Asynchronous Processing

Future → async callouts.  
Batch → large dataset processing (Deliveries, Orders).  
Queueable → notifications & chained jobs.  
Scheduled → reminders & daily updates.

**PHASE 6 – User Interface Development**

In this phase, the focus is on designing a user-friendly interface for the Delivery, Orders, Customers, and Loyalty Program modules. Salesforce’s Lightning framework and Lightning Web Components (LWC) are used to create a modern, interactive UI.

**1. Lightning App Builder**

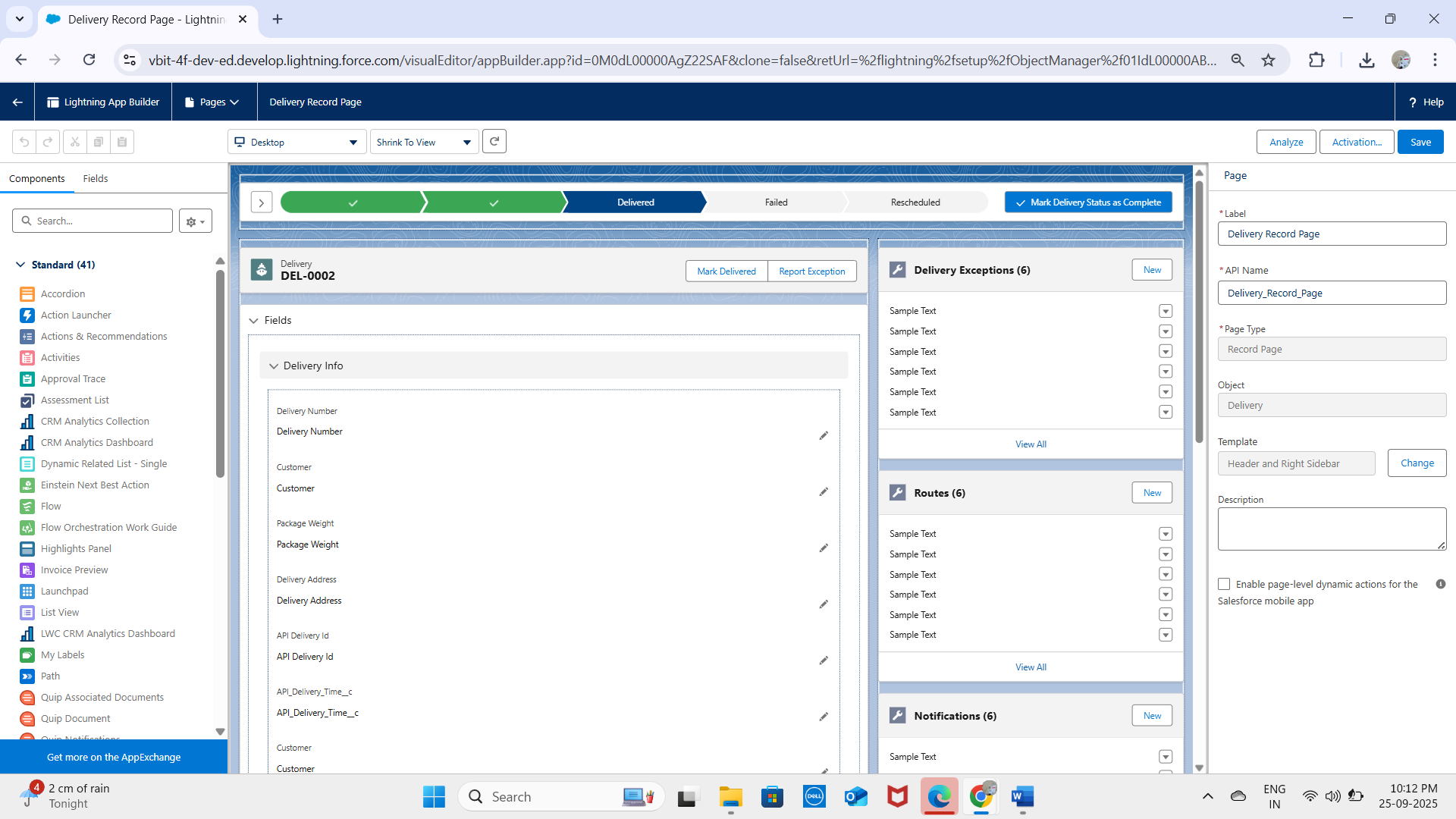
The Lightning App Builder was used to create custom pages for both Lightning Experience and the Salesforce mobile app.

* Drag-and-drop components to design pages without code.
* Example: Custom Order Management Dashboard displaying pending deliveries and loyalty points.

**2. Record Pages**

Record Pages were customized to show relevant details for each object:

* Delivery\_\_c Record Page: Shows Delivery Number, Status, Customer, and Scheduled Date.

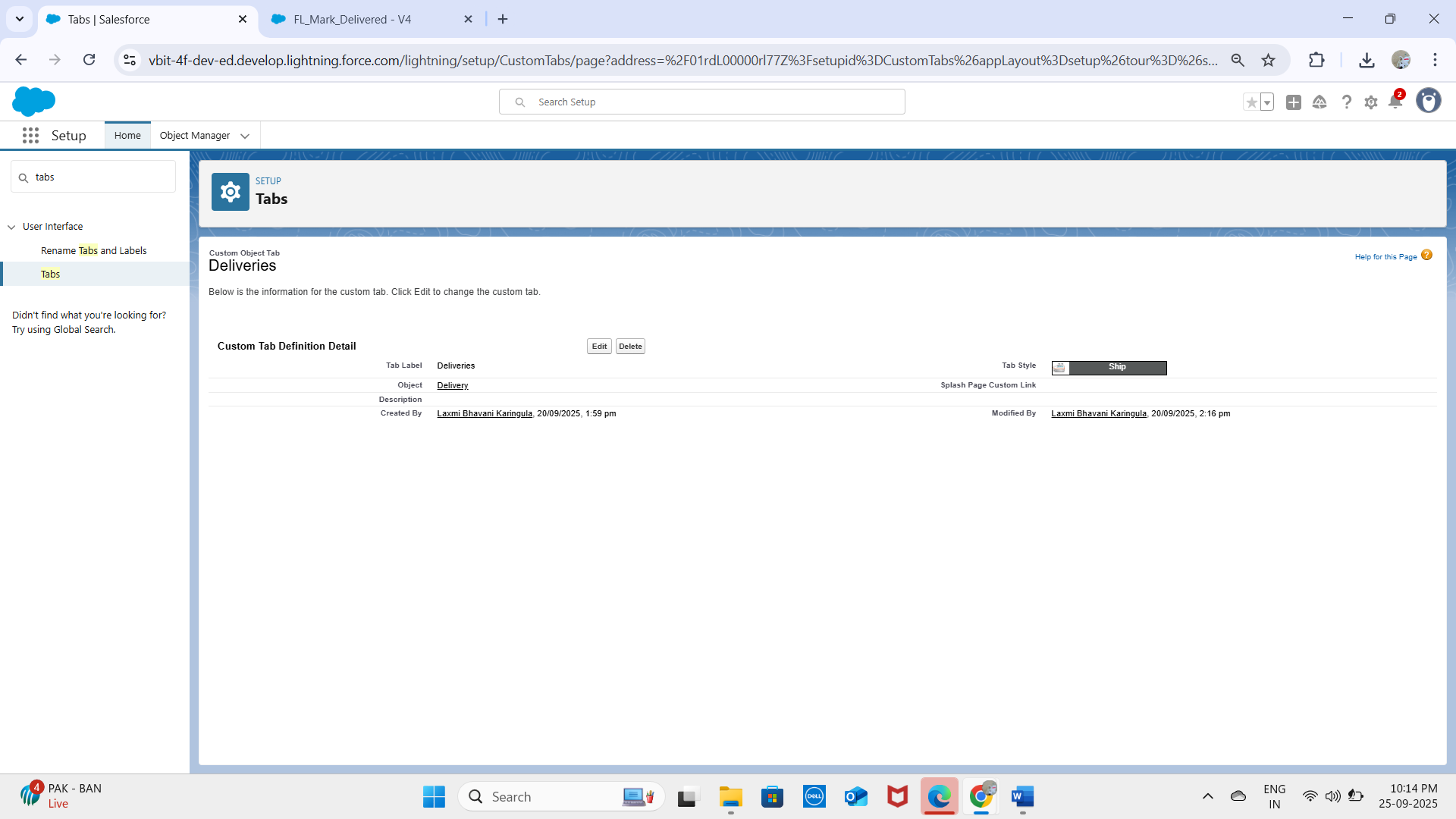


* Components like charts and related lists were added to enhance user experience.

**3. Tabs**

Custom tabs were created to improve navigation:

* Deliveries Tab – Access Delivery records.



**4. Home Page Layouts**

Custom home pages were designed for different profiles:

* Display key reports like Daily Deliveries, Pending Orders, and Loyalty Points Summary.
* Include quick links for creating new orders or deliveries.

**5. Utility Bar**

Configured the Utility Bar for quick access to:

* Notes for customer interactions.
* Recent deliveries and orders.
* Quick access to Loyalty Program components.

**6. Lightning Web Components (LWC)**

LWCs were used to create interactive UI components:

* Delivery Tracker LWC – Shows real-time delivery status.
* Order Summary LWC – Displays total orders and revenue dynamically

**7. Apex with LWC**

LWCs communicate with Apex classes to perform backend operations:

* Retrieve delivery and order records dynamically.
* Update loyalty points when orders are completed.
* Example: getCustomerLoyaltyPoints Apex method called from LWC.

**8. Events in LWC**

Custom events were implemented for component communication:

* Child-to-parent events to notify updates.
* Example: When an order is completed in Order Summary LWC, the Loyalty Points LWC is updated automatically.

**9. Wire Adapters**

Used wire adapters for reactive data fetching:

**@wire(getRecord, { recordId: '$recordId', fields: [NAME\_FIELD, STATUS\_FIELD] })**

**10. Imperative Apex Calls**

For conditional or dynamic operations, imperative Apex calls were used:

getCustomerLoyaltyPoints({ customerId: this.customerId })

.then(result => { this.points = result; })

.catch(error => { this.error = error; });

* Example: Fetch loyalty points only when an order is completed.

**11. Navigation Service**

Implemented the Navigation Service to navigate users seamlessly:

* Navigate from Orders List to Order Detail Page.
* Navigate from Customer Record to Loyalty Program Page.
* Provides consistent navigation without hardcoding URLs.

**✅ Outcome:**  
The user interface for the project now provides:

* A modern, interactive Lightning Experience.
* Easy navigation between Deliveries, Orders, Customers, and Loyalty modules.
* Real-time updates and notifications for users.
* Seamless integration between frontend LWCs and backend Apex logic.

**PHASE 7 – Integration & External Access**

In this phase, the focus is on integrating Salesforce with external systems and services for data exchange, automation, and real-time updates. Integration ensures that the Delivery, Orders, Customer, and Loyalty modules can communicate with other applications securely and efficiently.

**1. Named Credentials**

* Named Credentials provide a secure way to store authentication details for external systems.
* Avoids hardcoding usernames, passwords, or tokens in Apex.
* Example: Connecting to a Payment Gateway API to process customer payments for orders.

**2. External Services**

* Allows Salesforce to connect to APIs declaratively without heavy coding.
* Import API specifications (OpenAPI/Swagger) to automatically generate Apex actions.
* Example: Integrating a Delivery Tracking API to update delivery status in real-time.

**3. Web Services (REST/SOAP)**

* REST API: JSON-based, used for modern, lightweight integrations.
* SOAP API: XML-based, suitable for enterprise or legacy system integrations.
* Enables external systems to read, update, or delete Salesforce records.
* Example: External ERP system updates Order\_\_c records in Salesforce.

**4. Callouts**

* Outbound requests made from Salesforce to external systems via REST or SOAP.
* Requires Remote Site Settings or Named Credentials for security.
* Example: Sending Order details to an external delivery partner API for fulfillment.

**5. Platform Events**

* Supports event-driven communication within Salesforce or with external systems.
* Works using publish/subscribe model for real-time notifications.

Example: Publish a Delivery Completed Event when a delivery is done, triggering loyalty points updates in external systems.

**6. Change Data Capture (CDC)**

* Allows external systems to receive real-time notifications for record changes (create, update, delete, undelete).
* Useful for synchronizing Customer, Order, and Delivery data between Salesforce and external apps without frequent polling.

**7. Salesforce Connect**

* Enables Salesforce to access external data without importing it.
* Uses External Objects to display external records in real time.
* Example: Show Inventory Data from an external warehouse system in Salesforce Delivery records.

**8. API Limits**

* Salesforce enforces limits on API calls to ensure performance.
* Limits vary by edition and license type.
* Best practices: Use bulk APIs, caching, and event-driven integration for efficiency.

**9. OAuth & Authentication**

* OAuth is used for secure API authentication.
* Supports Single Sign-On (SSO) and secure external integrations.
* Example: A third-party logistics system accesses Salesforce data via OAuth tokens without exposing user passwords.

**10. Remote Site Settings**

* Define which external endpoints Salesforce is allowed to access.
* Required for HTTP callouts from Apex unless Named Credentials are used.
* Enhances security by preventing unauthorized external connections.

# PHASE 8 – Data Management & Deployment

Phase 8 focuses on managing Salesforce data efficiently and deploying customizations across environments while maintaining data quality, security, and integrity.

## 1. Data Import Wizard

- Salesforce’s guided tool for importing small to medium datasets (up to 50,000 records).  
- Supports standard objects (Accounts, Contacts, Leads) and custom objects (Delivery\_\_c, Order\_\_c, Loyalty\_Program\_\_c).  
- Field mapping ensures proper data alignment.  
- Example: Importing a bulk set of new Customer records or existing loyalty points.

## 2. Data Loader

- Client application for bulk operations (up to 5 million records).  
- Supports import, update, delete, and export.  
- Example: Updating Delivery Status for all deliveries after system migration.  
- More powerful than the Import Wizard and suitable for developers/advanced admins.

## 3. Duplicate Rules

- Maintain data quality by preventing or alerting duplicate records.  
- Uses matching rules on fields like Customer Email, Order Number, or Loyalty ID.  
- Options: block duplicates or allow with warnings.

## 4. Data Export & Backup

- Salesforce Data Export Service enables manual or scheduled backups in CSV format.  
- Essential for disaster recovery and compliance.  
- Example: Export all Orders, Deliveries, and Customer data weekly.  
- Third-party backup solutions can provide real-time data protection.

## 5. Change Sets

- Used to migrate metadata (custom objects, fields, flows, validation rules) between Salesforce environments.  
- Example: Deploy Delivery\_\_c object and its flows from Sandbox to Production.  
- Limitations: only works between connected orgs, supports only deployable metadata.

## 6. Unmanaged vs Managed Packages

- Unmanaged Packages: One-time metadata distribution, editable by recipients.  
 - Example: Sharing a custom Delivery Tracking solution for learning or internal use.  
- Managed Packages: Versioned, upgradeable, protects intellectual property.  
 - Example: ISV apps on AppExchange for loyalty program management.

## 7. ANT Migration Tool

- Command-line tool using Salesforce Metadata API.  
- Supports retrieving and deploying metadata programmatically.  
- Ideal for CI/CD automation.  
- Example: Automating deployment of Order and Delivery custom objects and flows.

## 8. VS Code & Salesforce DX (SFDX)

- Visual Studio Code: IDE for Salesforce development with extensions for Apex, LWC, and metadata.  
- SFDX CLI: Supports source-driven development, scratch orgs, version control, CI/CD pipelines.  
- Together, they enable advanced deployment, testing, and automation workflows.  
- Example: Using SFDX to deploy Phase 5 Apex classes, Phase 6 LWC components, and Phase 7 integration metadata.

## Outcome

With Phase 8 complete, the project now has:  
- Efficient data import/export and backup processes.  
- High data quality with duplicate prevention.  
- Smooth deployment workflow across Salesforce environments.  
- Support for CI/CD and source-driven development, ensuring maintainability and scalability.

**✅ Outcome:**  
With Phase 7 complete, the project now supports:

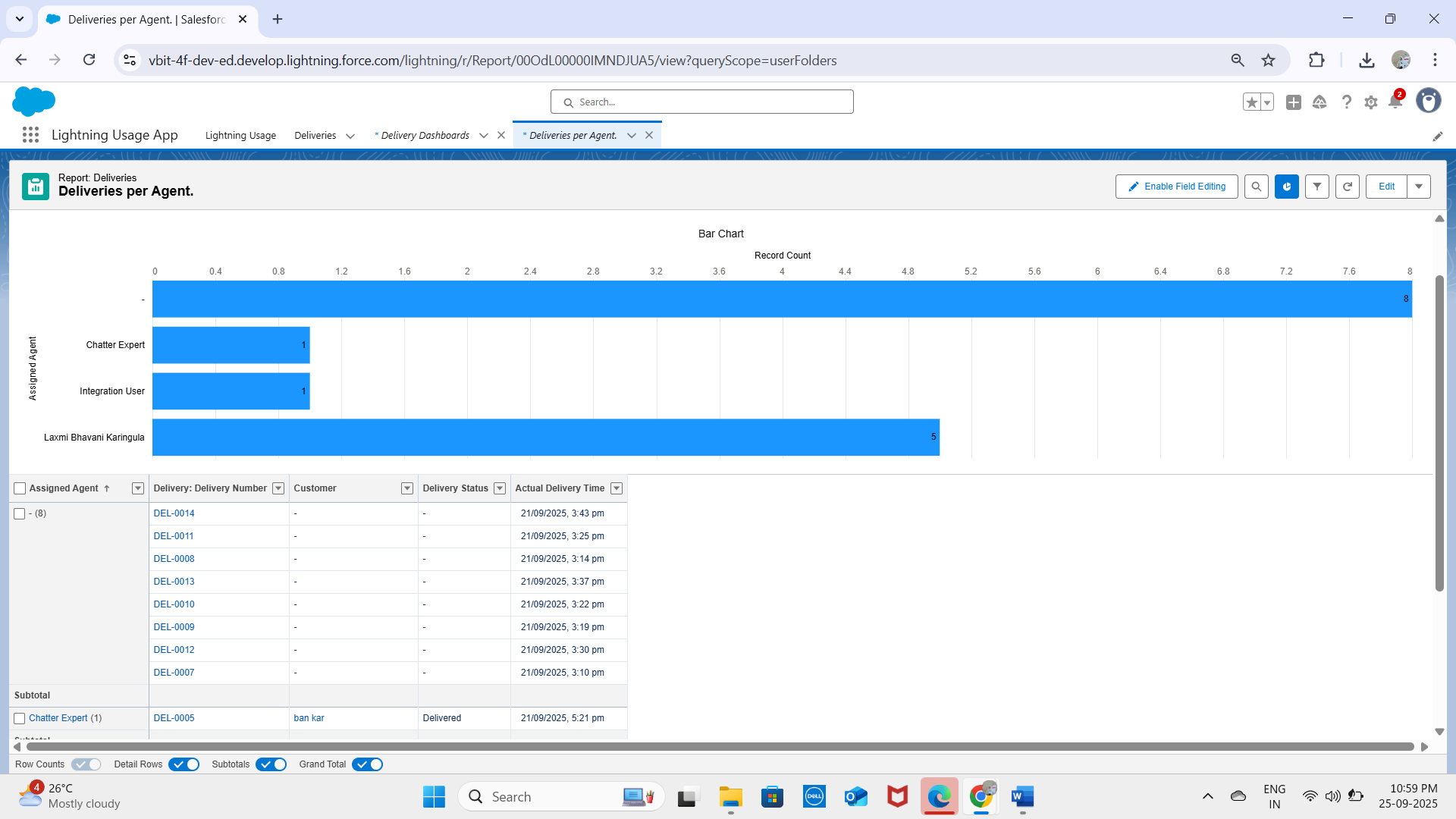
* Secure integration with payment gateways, delivery partners, and external systems.
* Real-time synchronization of Customer, Order, and Delivery data.
* Event-driven notifications for loyalty point updates.
* Scalable, secure, and maintainable external connections using Named Credentials, OAuth, and Remote Site Settings.

**PHASE 9 – Reporting, Dashboards & Security Review**

Phase 9 focuses on designing reports and dashboards for insights, and ensuring security measures are properly implemented for the Delivery, Orders, Customer, and Loyalty modules.

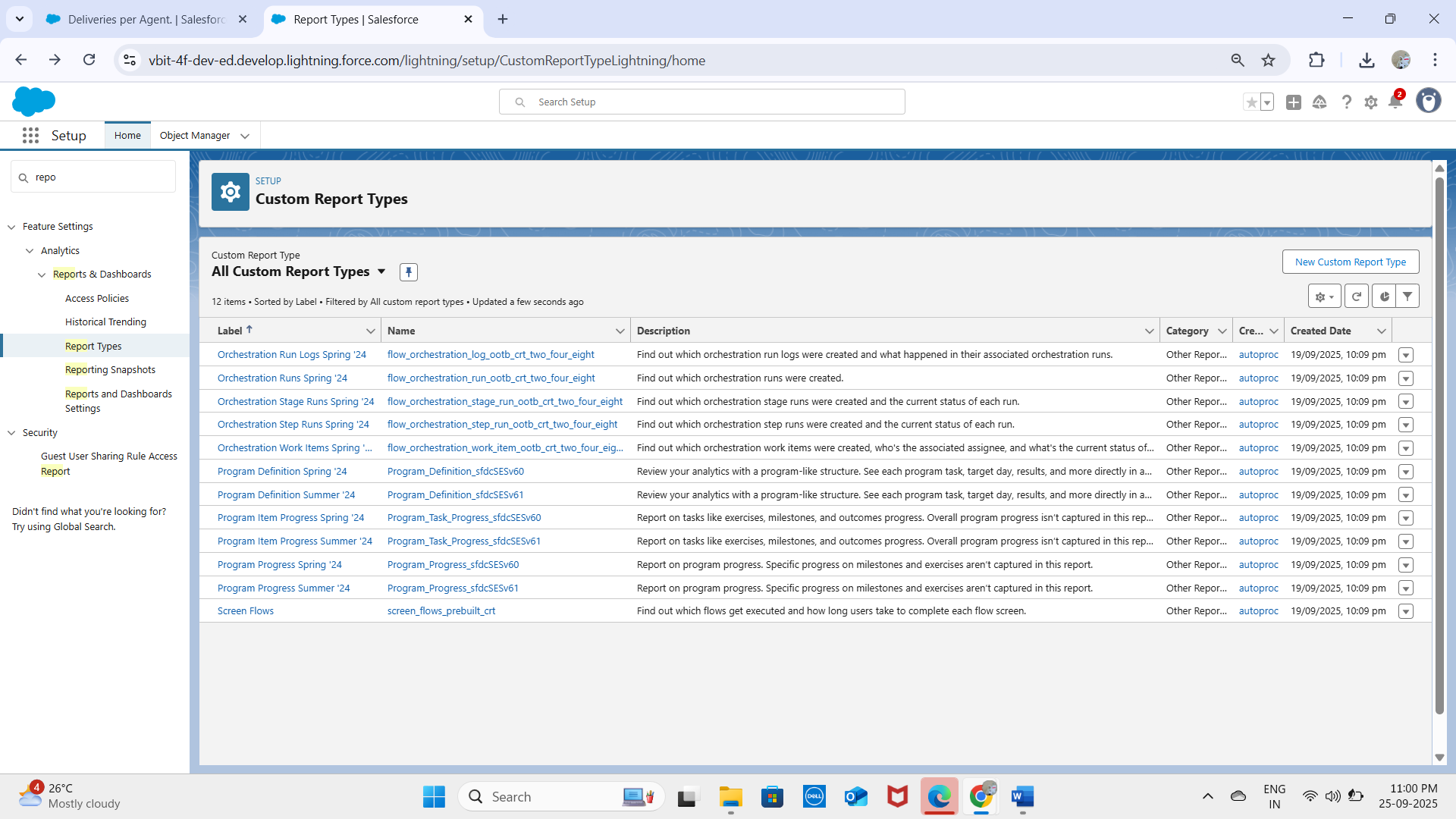
**1. Reports (Tabular, Summary, Matrix, Joined)**

* Tabular Reports → Simple list of records (like an Excel sheet). Best for tasks like mailing lists.
* Summary Reports → Grouped by rows with subtotals (e.g., sales by region).
* Matrix Reports → Grouped by rows and columns; best for comparing data in two dimensions (e.g., sales by product vs. region).
* Joined Reports → Combine multiple report blocks with different objects into a single view. Useful for comparing related datasets.



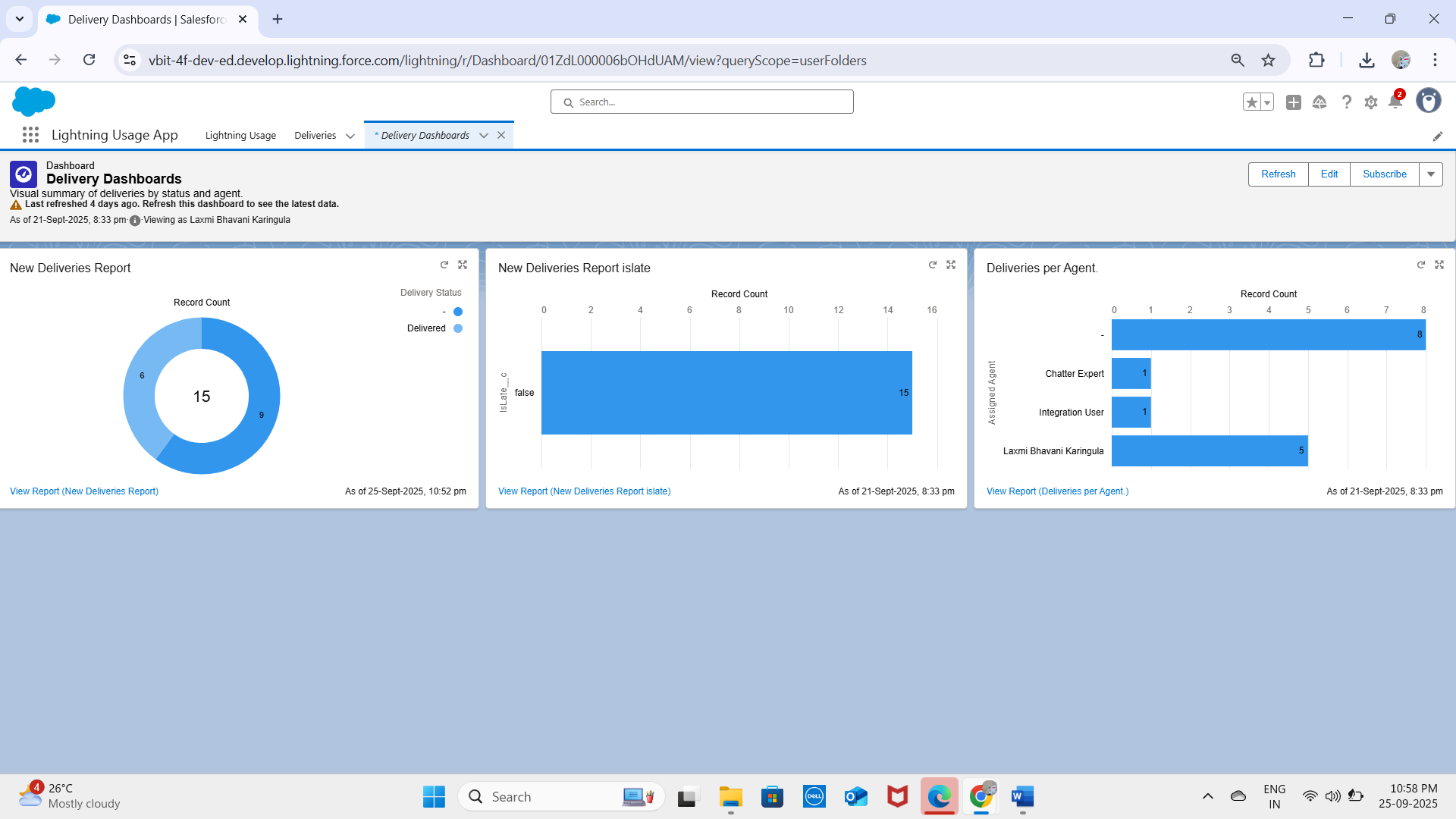
**2. Report Types**

* Standard Report Types → Auto-created by Salesforce for common relationships (e.g., Accounts with Contacts).
* Custom Report Types → Created by admins for specific reporting needs, including custom objects and complex relationships.



**3. Dashboards**

* Dashboards are visual representations of report data.
* Components include charts, tables, gauges, or metrics.
* Example: Dashboard displaying total Deliveries, Orders, and Loyalty Points.



**4. Dynamic Dashboards**

* Allow users to see data based on their own security settings rather than the dashboard creator’s.
* Example: A manager sees only their team’s deliveries, while an operator sees only assigned stations.

**5. Sharing Settings**

* Define default access (Organization-Wide Defaults) and record-level visibility.
* Admins can configure sharing rules for roles, groups, or territories.
* Ensures sensitive data is accessible only to authorized users.

**6. Field Level Security (FLS)**

* Controls access to individual fields on objects.
* Fields can be read-only, hidden, or editable based on profiles/permission sets.
* Protects sensitive information like payment details or customer PII.

**7. Session Settings**

* Define user session duration, multi-factor authentication requirements, and timeout warnings.
* Enhances security while maintaining usability.

**8. Login IP Ranges**

* Restrict logins to specific IP ranges to enhance security.
* Example: Internal users may only log in from corporate office IP addresses.
* Prevents unauthorized access from unknown networks.

**9. Audit Trail**

* Tracks configuration changes in Salesforce.
* Logs who made changes, what was changed, and when.
* Helps in compliance, troubleshooting, and monitoring unauthorized modifications.

**✅ Outcome:**  
With Phase 9 complete, the project ensures:

* Comprehensive reporting for Deliveries, Orders, Customers, and Loyalty Points.
* Dynamic dashboards for role-based data visibility.
* Strong security and monitoring via FLS, session settings, login restrictions, and audit trails.