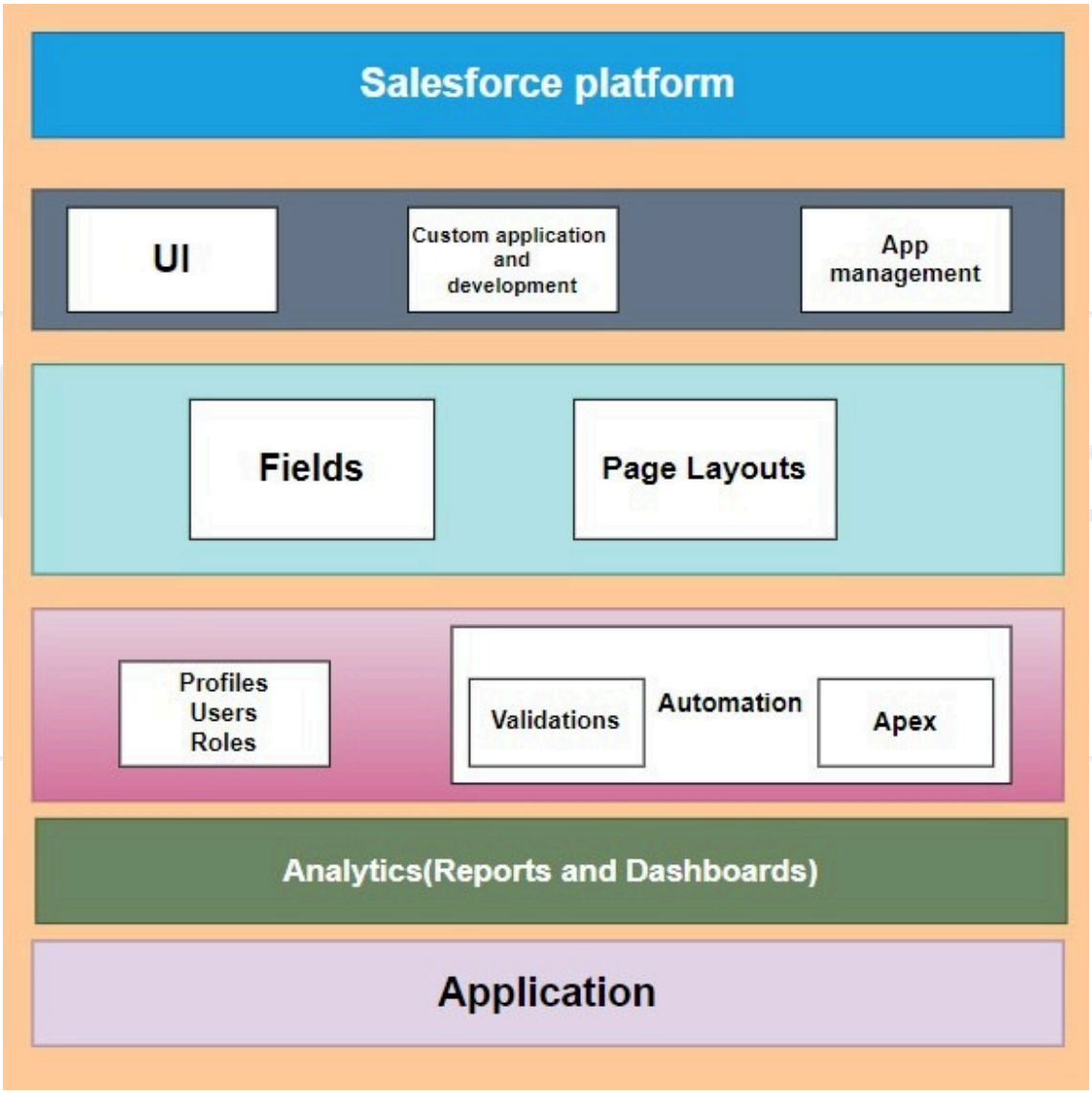
Dispatcher,Parcel delivery and tracking application

User Story : Streamline your parcel delivery operations with our Salesforce-powered application.
Effortlessly assign parcels to delivery agents, monitor real-time tracking, and enhance customer satisfaction. In today's fast-paced world, efficient parcel delivery and tracking services are essential for businesses and organizations of all sizes. Whether you're managing a small courier service or overseeing a large-scale logistics operation, our Salesforce-powered Dispatcher Parcel Delivery and Tracking Application is designed to meet your specific needs, optimize your processes, and enhance the overall customer experience.

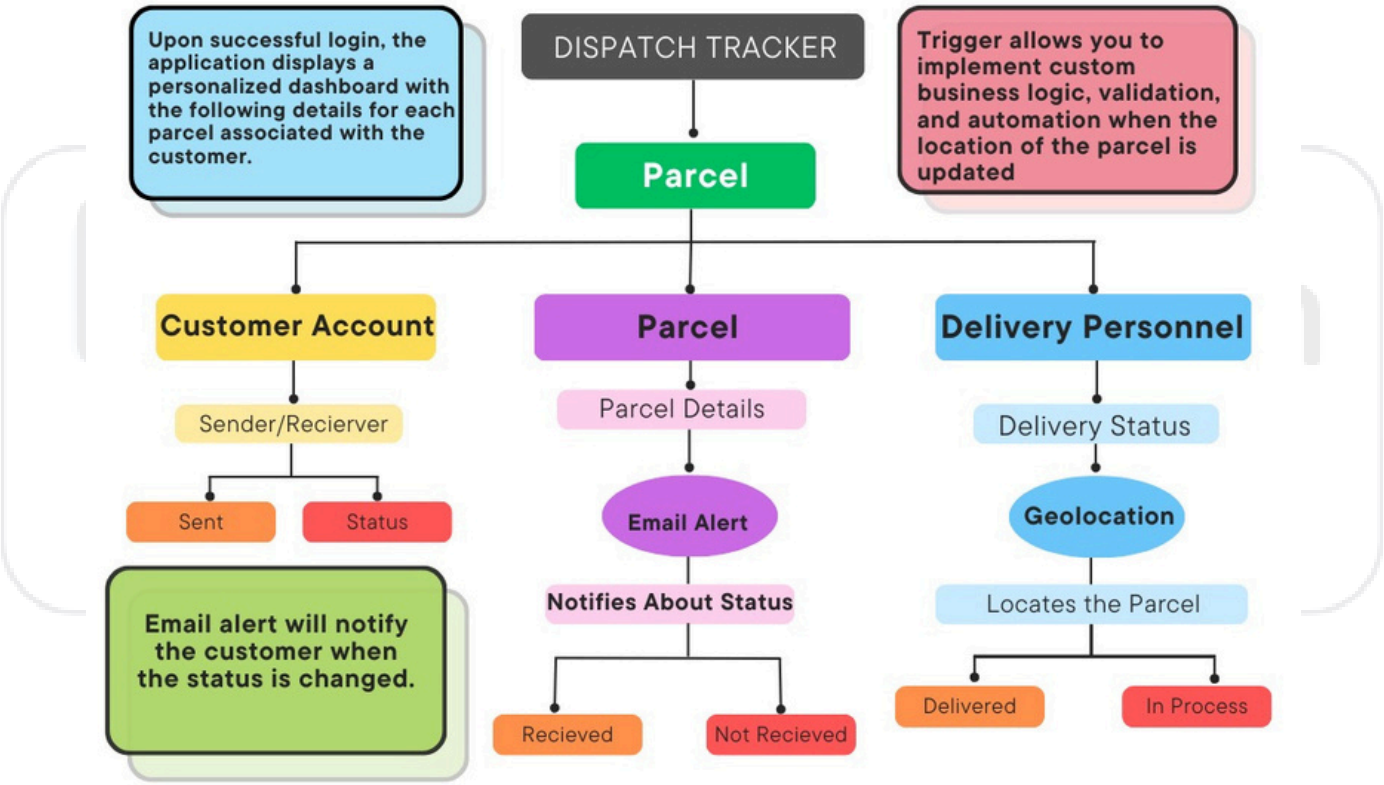
What you'll learn

1. RealTimeSalesforceProject
2. Object&RelationshipinSalesforce
3. Formulafields.
4. Apex
5. ApexTriggers
6. LightningWebComponents
7. ScheduleApex
8.Reports and Dashboards
9. Flows

Technical Architecture:



Project Flow:



Milestone 1- Create a Salesforce Developer Account :

What Is a Salesforce Developer Account?

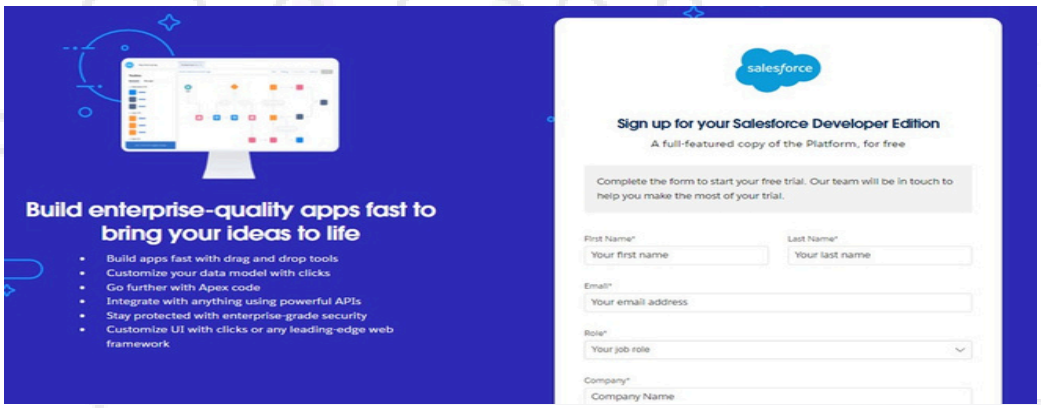
A Salesforce Developer Account, often referred to as a Developer Org, is a free Salesforce environment provided by Salesforce for developers to build, test, and experiment with Salesforce applications and customizations. It is a fully functional Salesforce instance, but it comes with some limitations and features tailored for development purposes.

Activity 1:

Creating Developer Account:

Creating a developer org in salesforce.

1. Gotohttps://developer.salesforce.com/signup
2. Onthesignupform,enterthefollowingdetails:



1) Firstname&Lastname

2) Email

3) Role:Developer

4) Company:CollegeName

5) County:India

6) PostalCode:pincode

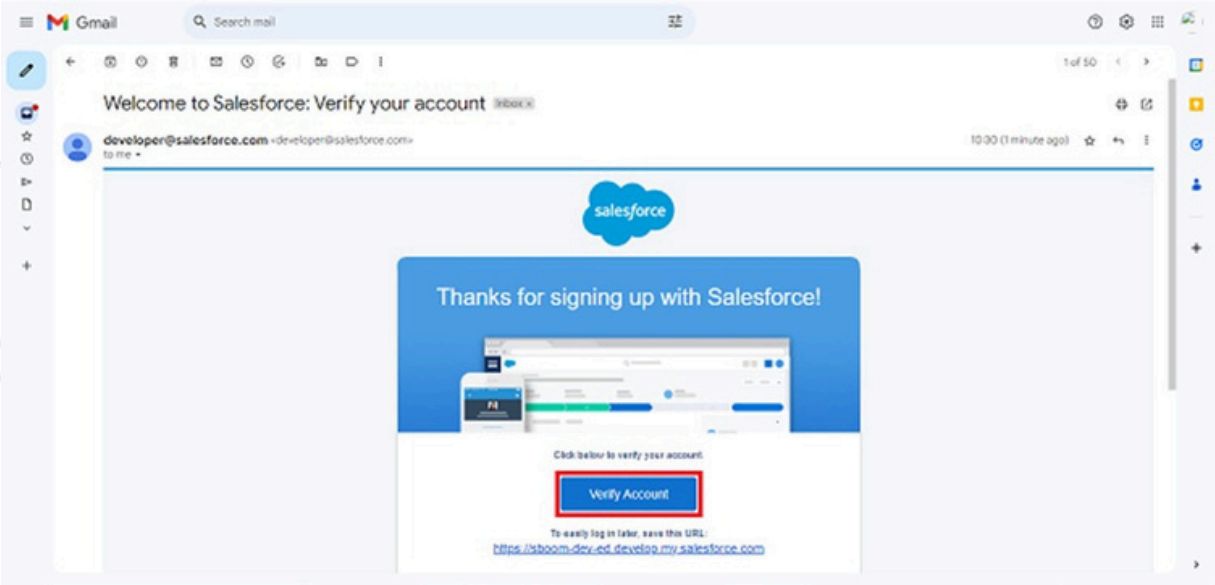
7) Username : should be a combination of your name and company

This need not be an actual email id, you can give anything in the format : username@organization.com

Click on sign me up after filling these.

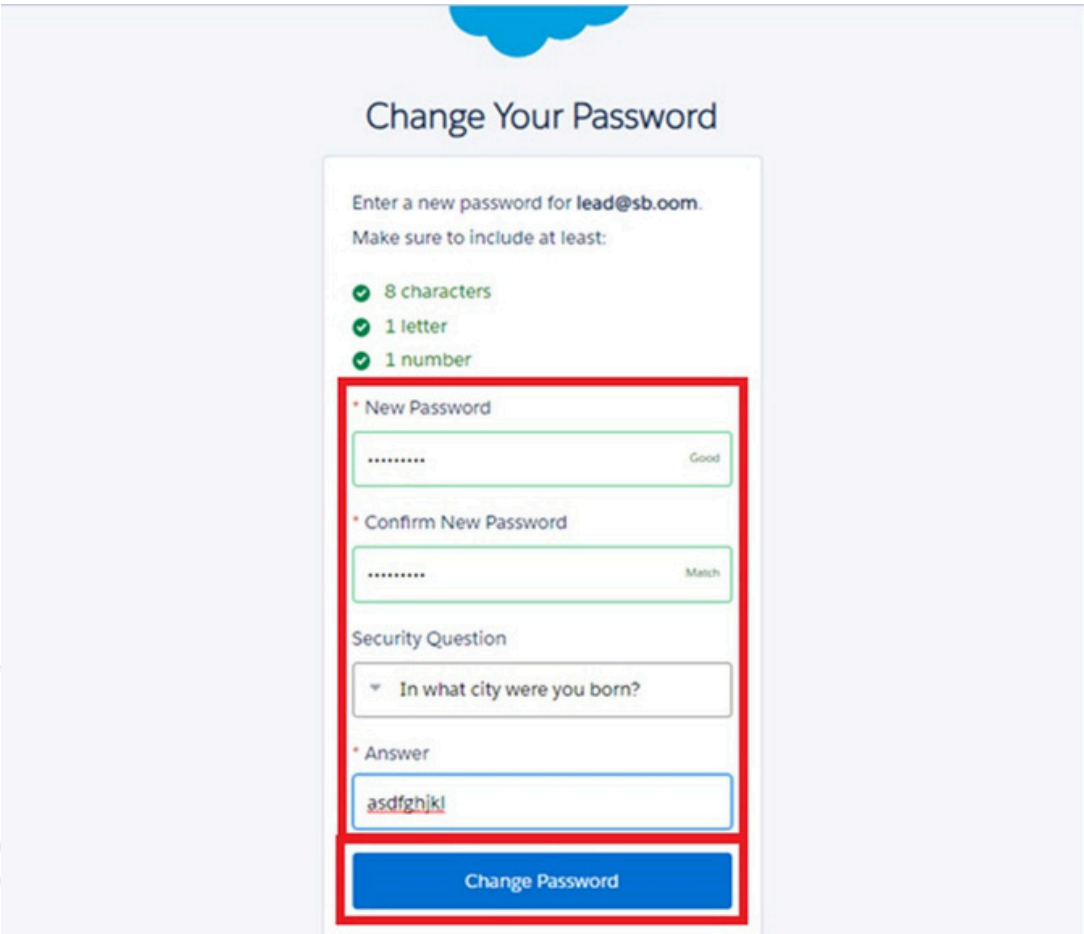
Activity 2: Account Activation:

1. Go to the inbox of the email that you used while signing up. Click on the verify account to activate your account. The email may take 5-10mins.

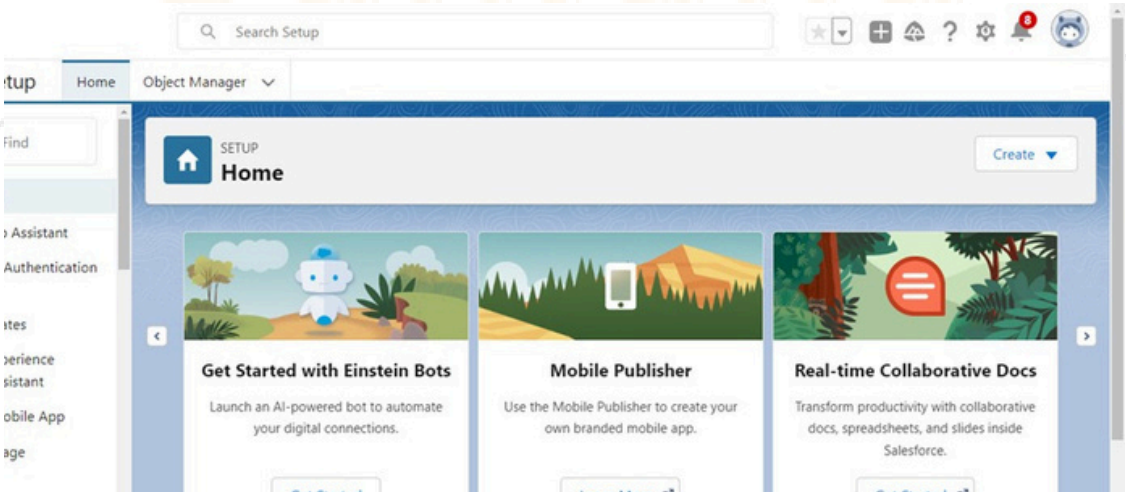


2. ClickonVerifyAccount.

3. Giveapasswordandanswerasecurityquestionandclickonchangepassword.



4. Then you will redirect to your salesforce setup page.



Milestone 2: Object

What are the objects required ? Salesforce objects are database tables that permit you to store data that is specific to an organization. It consists of fields (columns) and records (rows). Custom objects that are created by users for our case are Parcel object saves the essential information that a parcel holds i.e, parcel ID etc, similarly the other objects hold the information that is related to them, Customer Account and Delivery Personnel. They supply information that is essential to this project.

Activity -1

Create a custom object for Parcel

To create a custom object, follow these steps :

1. Click on the Gear Icon → From setup click on object manager
2. Click create, select custom object.
3. Fill in the label as "Parcel".
4. Fill in the plural label as "Parcel".
5. Record name : "Parcel Name"
6. Select the data type as "Text".
7. In the Optional Features section, select Allow Reports and Track Field History.
8. In the Deployment Status section, ensure Deployed is selected.
9. In the Search Status section, select Allow Search.
10. In the Object Creation Options section, select select these options:
Add Notes and Attachments related list to default page layout
Launch New Custom Tab Wizard after saving this custom object
11. Leave everything else as is, and click Save.

Activity -2

Create a custom object for Customer Account

To create a custom object, follow these steps :

1. Click on the Gear Icon → From setup click on object manager
2. Click create, select custom object.
3. Fill in the label as "Customer Account".
4. Fill in the plural label as "Customer Accounts".

5. Record name : "Customer Accounts"
6. Select the data type as "Text".
7. In the Optional Features section, select Allow Reports and Track Field History.
8. In the Deployment Status section, ensure Deployed is selected.
9. In the Search Status section, select Allow Search.
10. In the Object Creation Options section, select select these options:
Add Notes and Attachments related list to default page layout
Launch New Custom Tab Wizard after saving this custom object
11. Leave everything else as is, and click Save.

Activity - 3

Create a custom object for Delivery Personnel
1. Click on the Gear Icon → From setup click on object manager
2. Click create, select custom object.
3. Fill in the label as Delivery Personnel".
4. Fill in the plural label as "Delivery Personnels".
5. Record name : "Delivery Personnel"
6. Select the data type as "Text".
7. In the Optional Features section, select Allow Reports and Track Field History.
8. In the Deployment Status section, ensure Deployed is selected.
9. In the Search Status section, select Allow Search.
10. In the Object Creation Options section, select select these options:
Add Notes and Attachments related list to default page layout
Launch New Custom Tab Wizard after saving this custom object
11. Leave everything else as is, and click Save.

Milestone 3 : Tabs

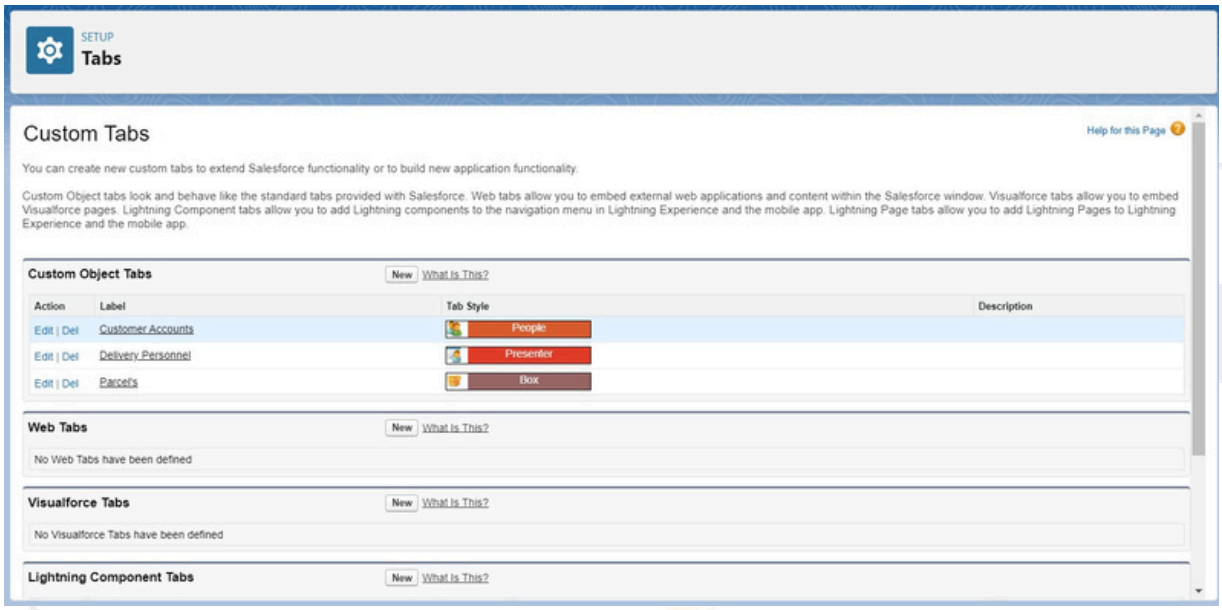
What is the requirement of Tab? Tab is a user interface element that allows users to navigate to different sections of the platform, such as Customer Account, Parcel and Delivery Personnel. Tabs are used to access custom objects and custom pages. These are located at the top of the screen and can be customized to fit the needs.

Activity - 1

How to create a tab

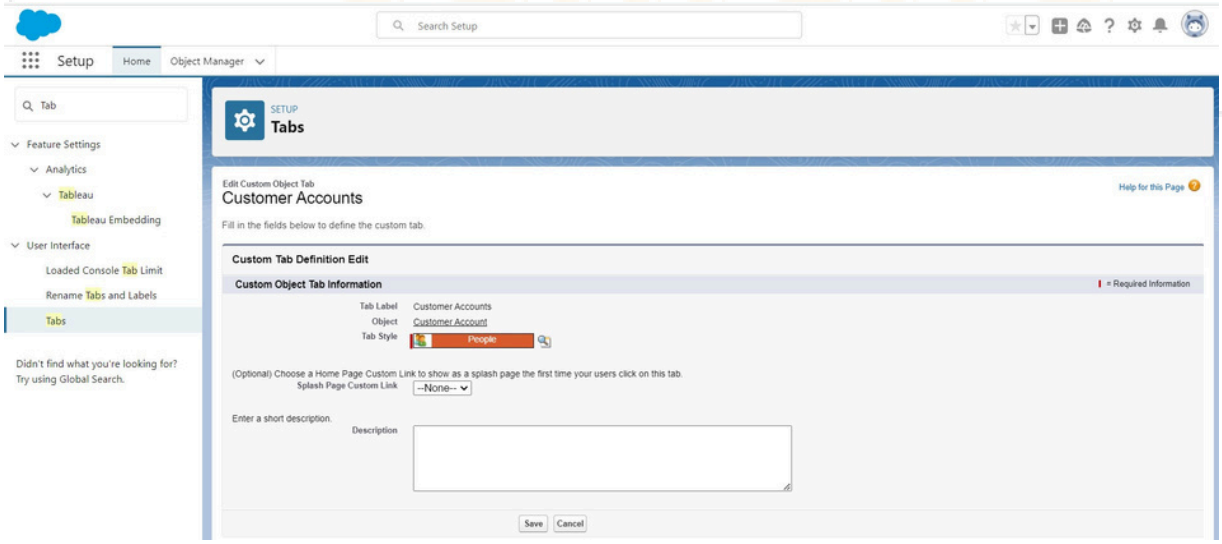
As we selected to launch a custom tab wizard in step 10, a custom tab wizard appears wherein We customize the look of the "Customer Account". object's tab. To do that :

1. To Create the tab → click on the gear Icon → then click Home button →In the quick find box search for the Tabs.



2. Then Click on the new button to create tab, in the Tab style you can select whatever you want to select

3. Than select the Customer Account Object and select the Icon from the search button and click next next and save



For creating the Remaining tabs follow the above steps

a. For creating a tab for Parcel Value Tab follow the above steps which are being created in step1 for customer account.

b. For creating a tab for Delivery Personnel Tab follow the above steps which are being created in step1 for customer account.

Milestone 4:Fields & Relationships

Fields represent the data stored in the columns of a relational database. It can also hold any valuable information that you require for a specific object. Hence, the overall searching, deletion, and editing of the records become simpler and quicker.

Types of Fields

- Standard Fields
- Custom Fields

Standard Fields:

As the name suggests, the Standard Fields are the predefined fields in Salesforce that perform a standard task. The main point is that you can't simply delete a Standard Field until it is a non-required standard field. They are

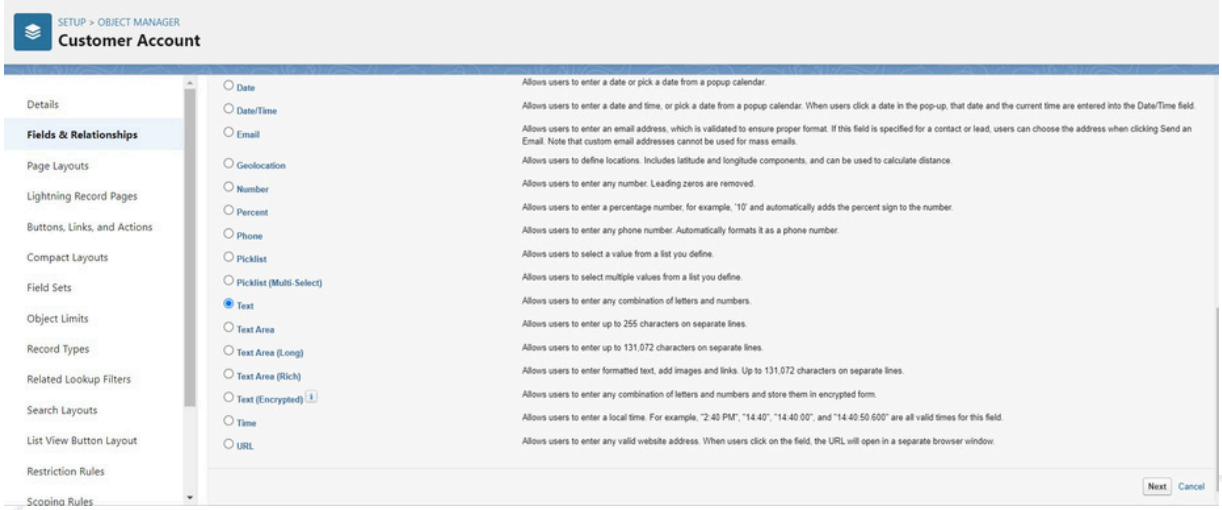
- Created By
- Owner
- Last Modified

Custom Fields:

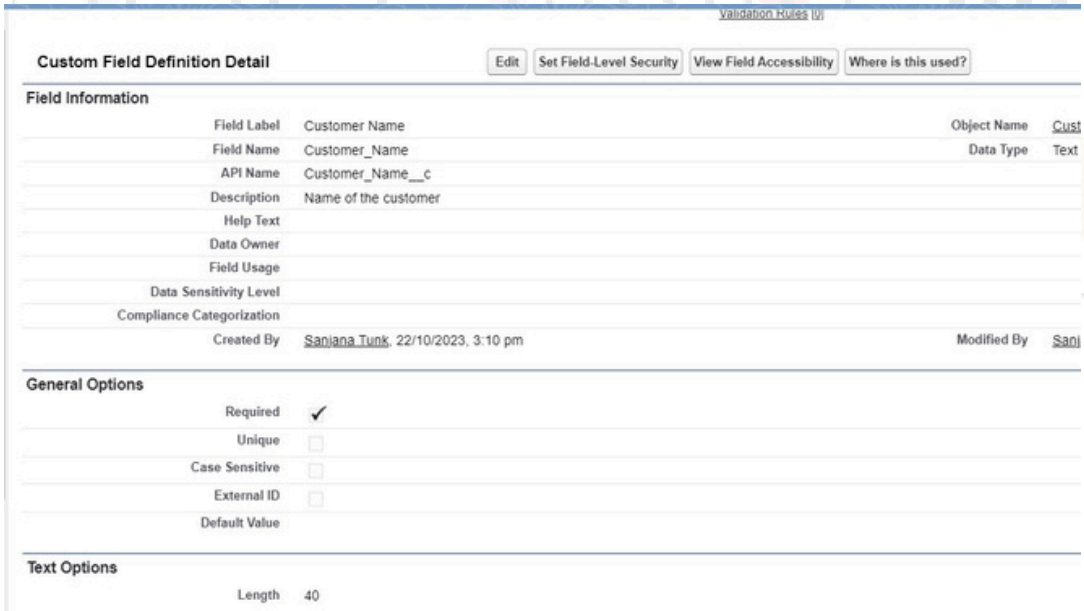
Custom Fields are highly flexible, and users can change them according to requirements. Moreover, each organizer or company can use them if necessary. It means you need not always include them in the records, unlike Standard fields. Hence, the final decision depends on the user, and he can add/remove Custom Fields of any given form.

Activity -1

1. Go to setup → click on Object Manager → type object name in search bar → click on the object→ "Customer Account" → Field and Relationship→ then click on new →Field Data Type(Text)



2. Click On Next→ Field Label →Customer Name→ next→next→Save. Always require a value in this field in order to save a record.



3. Follow the above steps and create the Remaining Fields.
   - CustomerContact(Phone)-Alwaysrequireavalueinthisfieldinordertosavearecord
   - CustomerEmailId(Email)-Alwaysrequireavalueinthisfieldinordertosavearecord
   - Password (Password) - Make sure to check unique checkbox-(Treat "ABC" and "abc" as duplicate values (case insensitive)) - Length is 30.
   - CustomerAddress(Geolocation)-DecimalPlaces15-Alwaysrequireavalueinthisfieldin order to save a record.

4. Click On Save & New→ Lookup Relationship → Select Parcel from related to list→ next→Field label Parcel→Save.

- DeliveryAddress(Geolocation)-DecimalPlaces15-Alwaysrequireavalueinthisfieldin order to save a record.
- Parcel Id (Text) - Make sure to check unique checkbox (case insensitive) -Always require a value in this field in order to save a record - Length is 16.
- DeliveryDate(Date)-Alwaysrequireavalueinthisfieldinordertosavearecord.
- PickupDate(Date)-Alwaysrequireavalueinthisfieldinordertosavearecord.
- ParcelStatus(Picklist)-Alwaysrequireavalueinthisfieldinordertosavearecordina separate line to be entered - In Transit: The parcel is currently being shipped or transported.
  Out for Delivery: The parcel is out for delivery and is expected to be delivered to the recipient.
  Delivered: The parcel has been successfully delivered to the recipient.
  On Hold: The delivery of the parcel has been temporarily delayed or put on hold.
  Scheduled for Pickup: The parcel is scheduled to be picked up by the carrier.
  Failed Delivery Attempt: An attempt to deliver the parcel was unsuccessful.
  Returned: The parcel has been returned to the sender or a return has been initiated.

4. Click On Save & New→ Lookup Relationship → Select Delivery Personnel from related to list→ next→Field label Delivery Personnel→Save.

Activity -3

1. Go to setup → click on Object Manager → type object name in search bar → click on the object→ “Delivery Personnel” → Field and Relationship→ then click on new →Field Data Type(Text)

2. Click On Next→ Field Label→ Delivery Personnel Name→ next→next→Save. Always require a value in this field in order to save a record.

3. Follow the above steps and create the Remaining Fields.
- ContactNumber(Phone)-Alwaysrequireavalueinthisfieldinordertosavearecord
- EmailId(Email)-Alwaysrequireavalueinthisfieldinordertosavearecord
- Route(Geolocation)-DecimalPlaces15-Alwaysrequireavalueinthisfieldinordertosavea record.

4. Click On Save & New→ Lookup Relationship → Select Parcel from related to list→ next→Field label Parcel→Save.

SETUP > OBJECT MANAGER

Delivery Personnel

- Details
- **Fields & Relationships**
- Page Layouts
- Lightning Record Pages
- Buttons, Links, and Actions
- Compact Layouts
- Field Sets
- Object Limits
- Record Types
- Related Lookup Filters
- Search Layouts
- List View Button Layout

Geolocation	be used to calculate distance.
Number	Allows users to enter any number. Leading zeros are removed.
Percent	Allows users to enter a percentage number, for example, '10' and automatically adds the percent sign to the number.
Phone	Allows users to enter any phone number. Automatically formats it as a phone number.
Picklist	Allows users to select a value from a list you define.
Picklist (Multi-Select)	Allows users to select multiple values from a list you define.
Text	Allows users to enter any combination of letters and numbers.
Text Area	Allows users to enter up to 255 characters on separate lines.
Text Area (Long)	Allows users to enter up to 131,072 characters on separate lines.
Text Area (Rich)	Allows users to enter formatted text, add images and links. Up to 131,072 characters on separate lines.
Text (Encrypted)	Allows users to enter any combination of letters and numbers and store them in encrypted form.
Time	Allows users to enter a local time. For example, "2:40 PM", "14:40", "14:40:00", and "14:40:50.600" are all valid times for this field.
URL	Allows users to enter any valid website address. When users click on the field, the URL will open in a separate browser window.

Next Cancel

SETUP > OBJECT MANAGER

Delivery Personnel

- Details
- **Fields & Relationships**
- Page Layouts
- Lightning Record Pages
- Buttons, Links, and Actions
- Compact Layouts
- Field Sets
- Object Limits
- Record Types
- Related Lookup Filters
- Search Layouts
- List View Button Layout

Delivery Personnel Custom Field

Delivery Personnel Name

Back to Delivery Personnel

Validation Rules [0]

Custom Field Definition Detail

Edit | Set Field-Level Security | View | Where is this used?

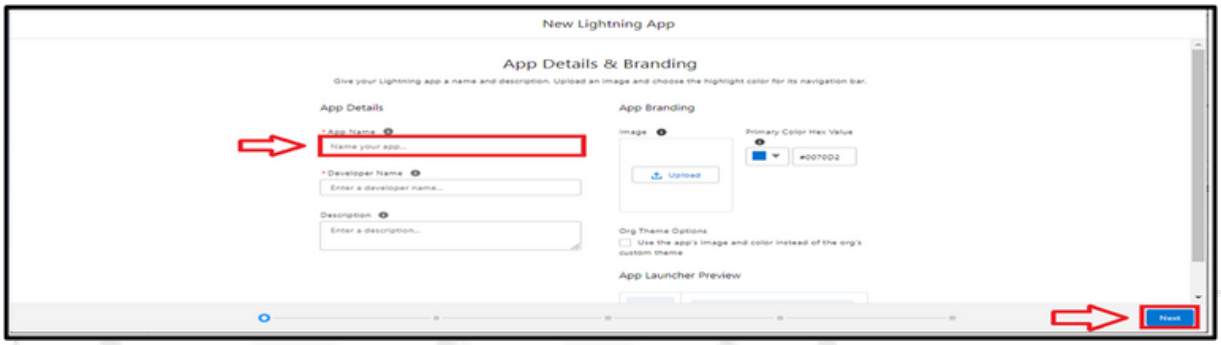
Field Information

Field Label	Delivery Personnel Name	Object Name	
Field Name	Delivery_Personnel_Name	Data Type	
API Name	Delivery_Personnel_Name__c		
Description			
Help Text			
Data Owner			
Field Usage			
Data Sensitivity Level			
Compliance Categorization			
Created By	Sanjana Tunk, 22/10/2023, 3:20 pm	Modified By	

General Options

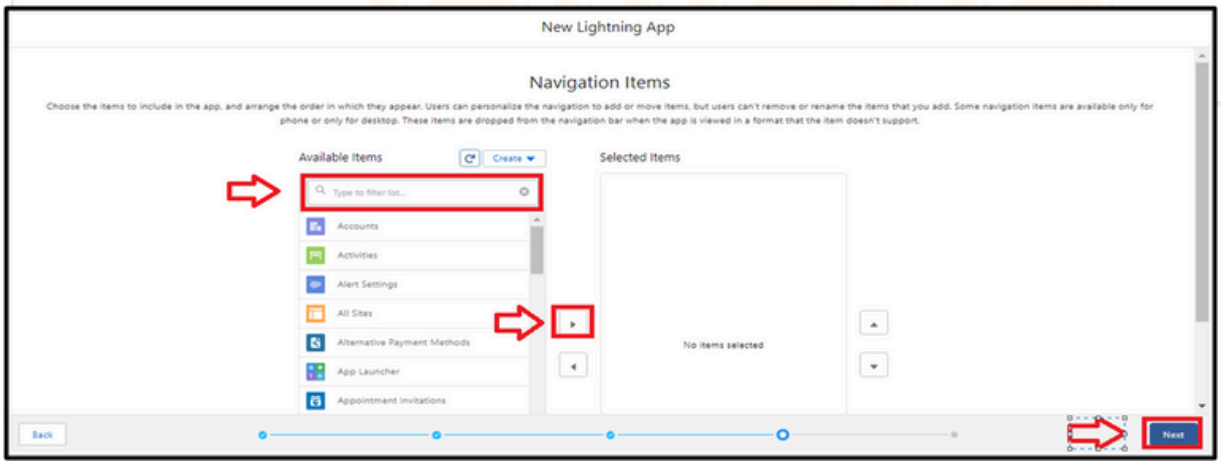
Required ✓

3. Utility Items keep it as default → Next → (Add Navigation Items)(add tabs Customer Accounts, Parcel's Delivery Personnel) → Next → (Add User Profile) Add System Administrator, Salesforce platform user, Standard User → Next.



4. To Add Navigation Items:
Select the items from the search bar and move it using the arrow button → Next. select all the tabs which you have created

5. To Add User Profiles:
Search profiles in search bar → click on the arrow button & select Standard user, standard Platform user & System Admin Profile→ save & finish.



Milestone 6: Profile

A profile is a group/collection of settings and permissions that define what a user can do in salesforce. Profile controls “Object permissions, Field permissions, User permissions, Tab settings, App settings, Apex class access, Visualforce page access, Page layouts, Record Types, Login hours & Login IP ranges. You can define profiles by the user's job function. For example System Administrator, Developer, Sales Representative.

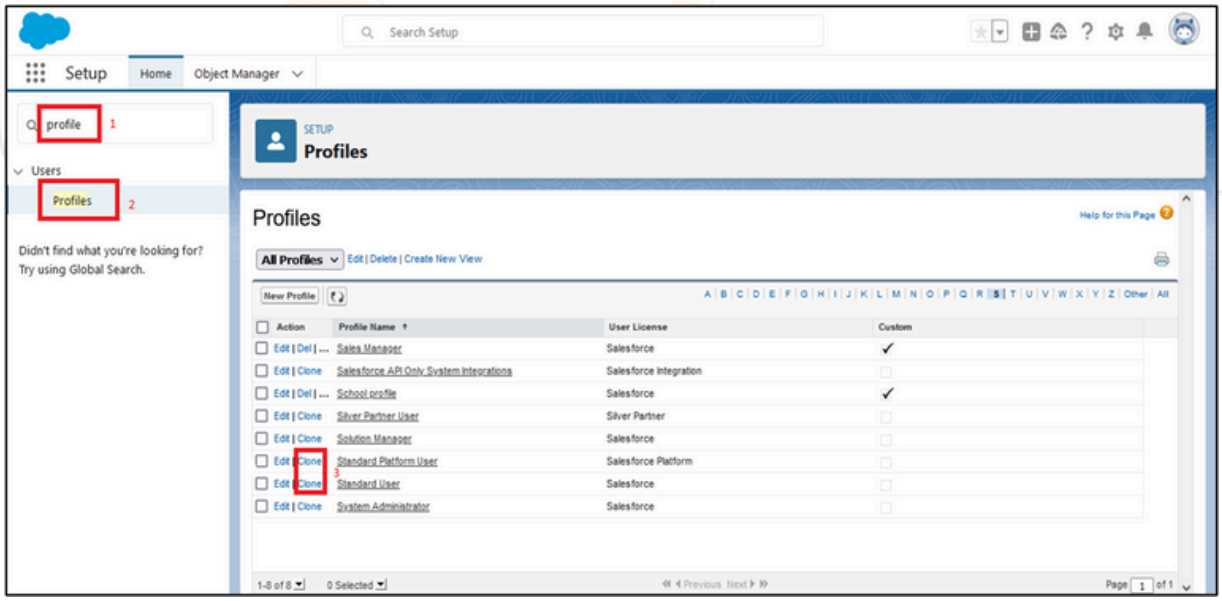
Custom Profiles:

- Custom ones defined by us i.e; Delivery Personnel, Customer etc.
- They can be deleted if there are no users assigned with that particular one

Activity-1

To create a new profile:

1. Go to setup → type profiles in quick find box → click on profiles → clone the desired profile (standard platform user is pref) and clone that profile



2. Enter a Profile Name(Delivery Personnel) And click on Save
3. Click on the new created profile
4. While still on the profile page, then click Edit.

As a new administrator, you perform user management tasks like creating and editing users, resetting passwords, granting permissions, configuring data access, and much more. In this unit, you will learn about users and how you add users to your Salesforce org. Every user in Salesforce has a user account. The user account identifies the user, and the user account settings determine what features and records the user can access. Each user account contains at least the following:

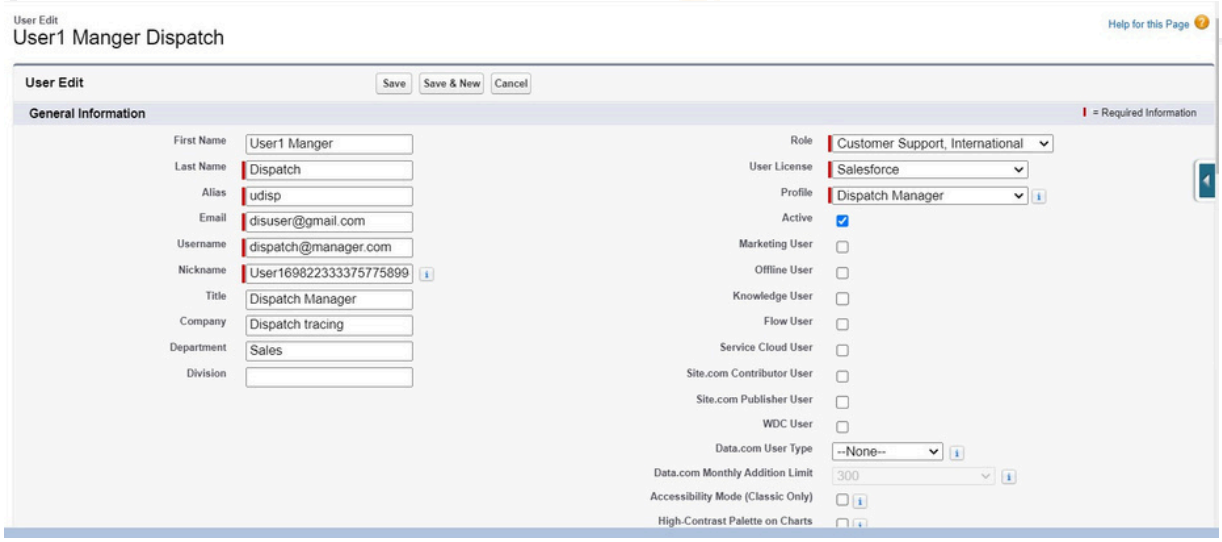
- Username
- EmailAddress
- User'sFirstName(optional)
- User'sLastName
- Alias
- Nickname
- License
- Profile
- Role(optional)

Now create the users with 3 profiles which we have created.

Activity - 1

User1 Dispatch Manager

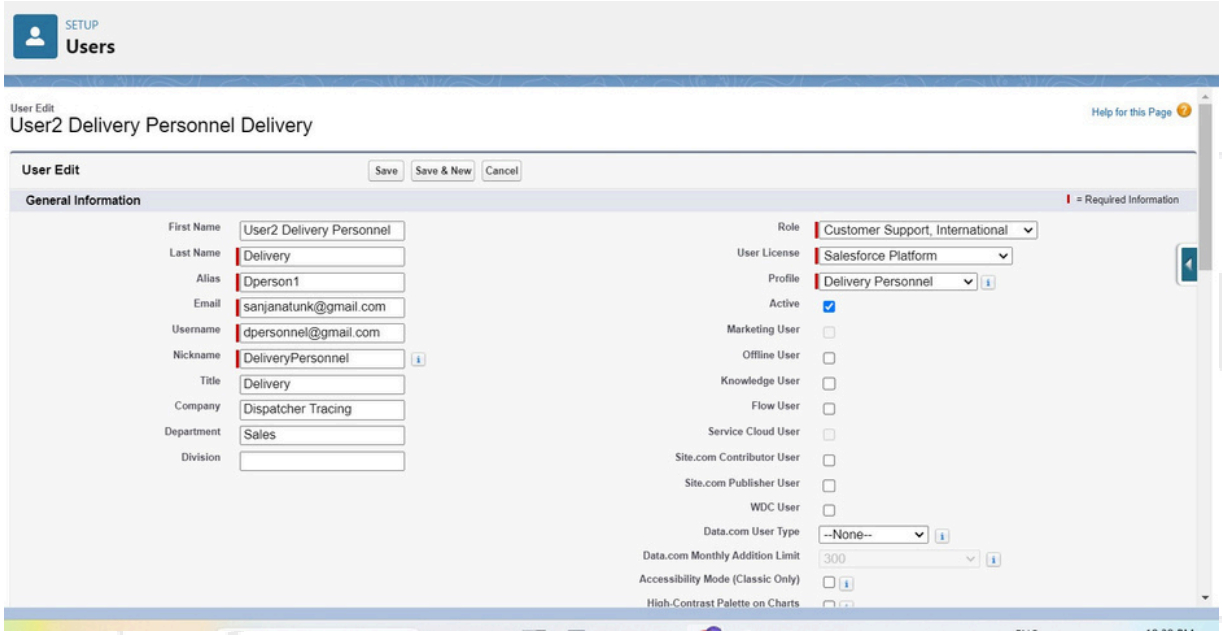
1.Go to the Gear Icon→ Click the setup→click on the home button and in the quick find box search for the user →Click on the user→ then click on new →fill the fields.



Activity - 2

User2 Delivery Personnel

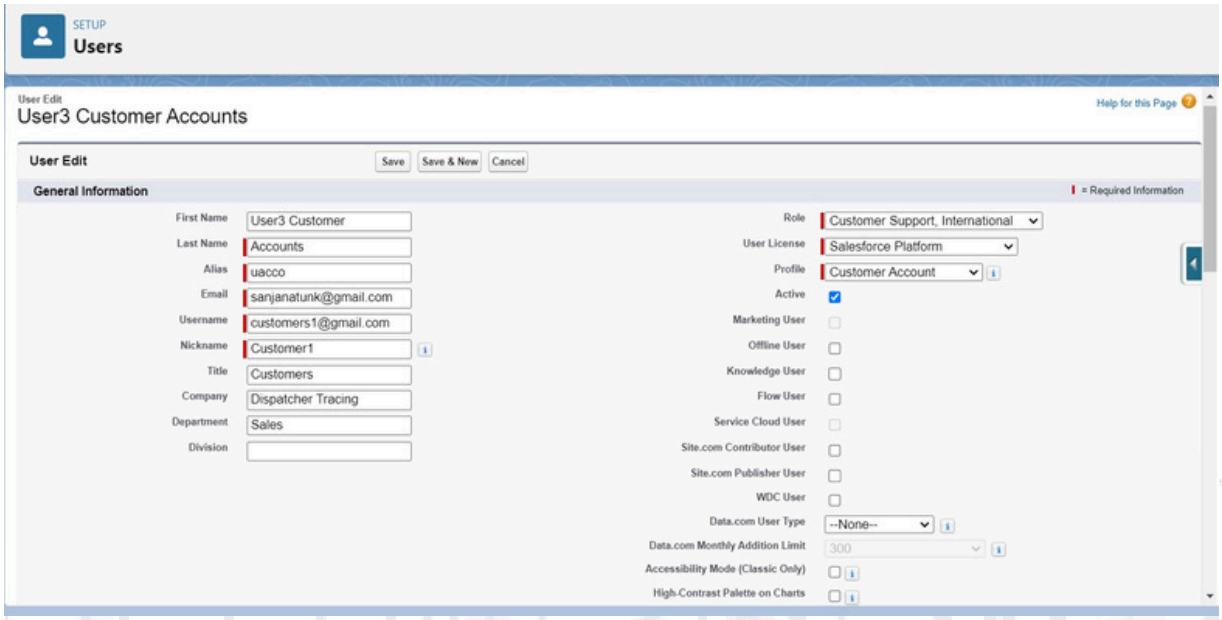
1.Go to the Gear Icon→ Click the setup→click on the home button and in the quick find box search for the user →Click on the user→ then click on new →fill the fields.



Activity - 3

User3 Customer Account

1.Go to the Gear Icon→ Click the setup→click on the home button and in the quick find box search for the user →Click on the user→ then click on new →fill the fields.



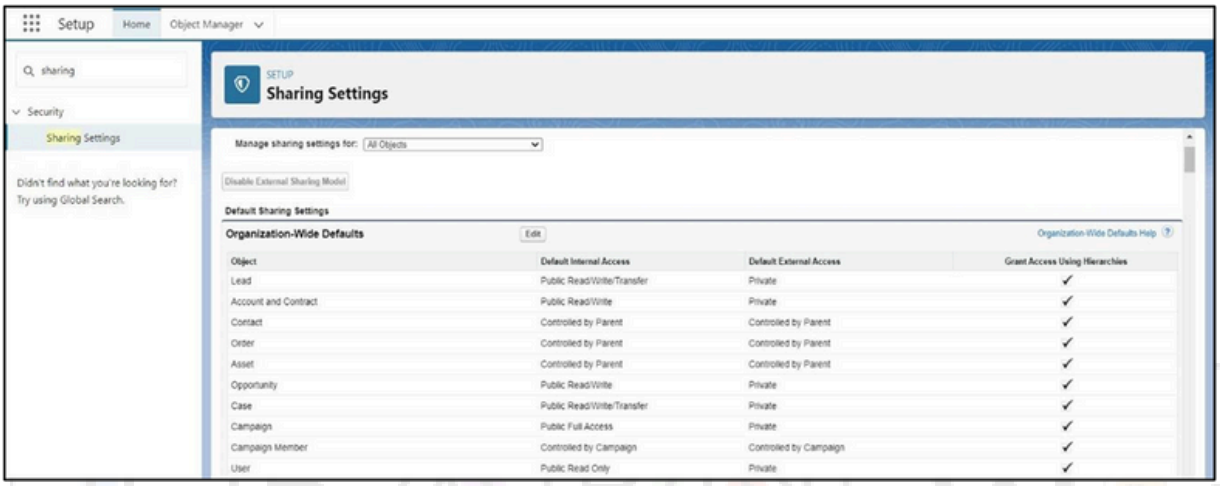
Milestone 8-OWD(Organization Wide Default)

OWD in a Parcel Dispatcher project is critical for establishing a secure baseline of data access. They help ensure that sensitive parcel information is appropriately protected, and access is granted based on organizational needs and compliance requirements. Primarily, there are four levels of access that can be set in Salesforce OWD and they are ● Public Read/Write/Transfer (only available of Enquiry and Cases) ● Public Read/Write ● Public Read/Only ● Private With parcel-related data stored in a custom object (e.g., "Parcels"), setting the OWD to Private ensures that users can only access the records they own or have been granted access to explicitly. If sender and recipient information is stored in standard or custom objects, OWDs help control access. Depending on the business requirements, you may set different OWDs for objects or fields related to tracking and delivery status.

Activity-1

Create OWD Setting

1. Setup, use the Quick Find box to find Sharing Settings.
2. Click Edit in the Organization-Wide Defaults area.
3. For each object, select the default access you want to give everyone.
4. For Every custom object give private as a record level security



Milestone 9-Reports

Reports give you access to your Salesforce data. You can examine your Salesforce data in almost infinite combinations, display it in easy-to-understand formats, and share the resulting insights with others. Before building, reading, and sharing reports, review these reporting basics.

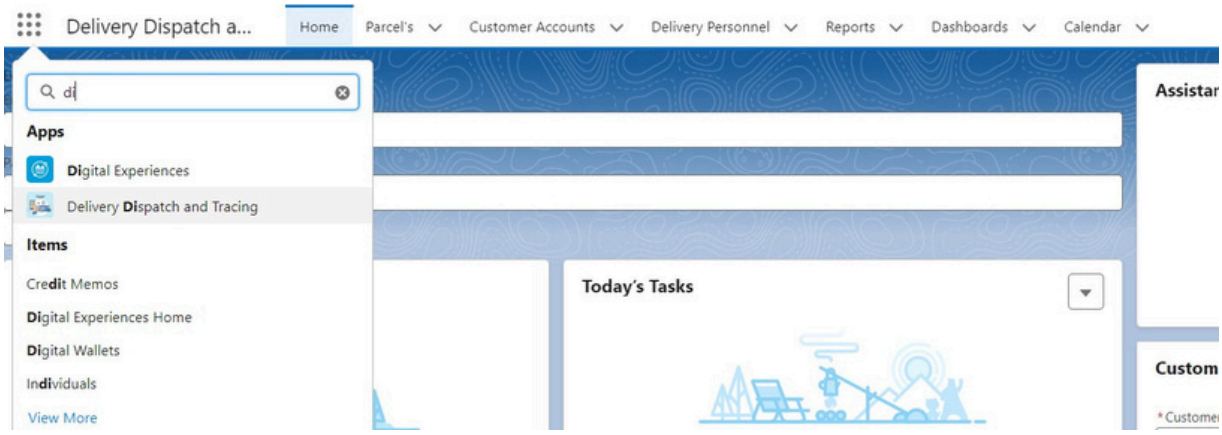
Types of Reports in Salesforce

- Tabular
- Summary
- Matrix
- JoinedReports

Activity-1

Create the Report

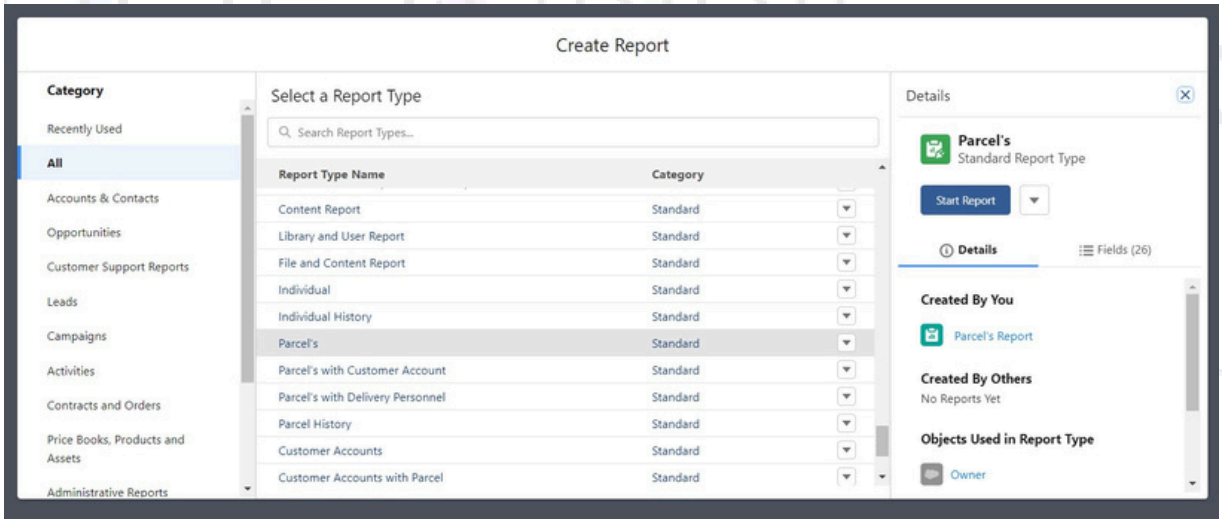
1. Go to the gear icon → click on the setup→click on the home button→ in the the quick find box search for app manager select the App and go to the navigation→ There select the Report and include in the app → than go to App There you will find the Tab



2. Click create folder → Public Reports→ New Report →Create the Report → Select the Parcel's Report → columns→ Parcel Name→Receivers Name.

Select Start Report:

3. For to the reports and select the → Parcel's



4. Select the following option for the Rows and columns.

Notification Handling:

- If the status meets certain criteria, the trigger adds the parcel to parcelsToUpdate and sends email notifications using the handler (ParcelNotificationClass).

Field Update:

- The trigger updates the ParcelStatus__c field for certain statuses.

ScheduledNotification:

- The scheduler class (ParcelNotificationScheduler) simulates sending a notification for a delivered parcel every weekday at 8:00 AM.

Activity- 1

Defining the logic for automating actions based on trigger events, Apex class to handle various parcel stages, including "Pickup," "In Transit," "Out for Delivery," "Delivered," "On Hold," "Failed Delivery Attempt," and "Returned." It includes methods for sending notifications for these stages

```
public class ParcelNotificationClass {
    public static void sendNotification(String recipientEmail, String parcelId, String status) {
        // Compose the email message based on the parcel status
        Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
        email.setToAddresses(new String[]{recipientEmail});

        if (status == 'Pickup') {
            email.setSubject('Parcel Scheduled for Pickup');
            email.setPlainTextBody('Your parcel with ID ' + parcelId + ' is scheduled for pickup.');
        } else if (status == 'In Transit') {
            email.setSubject('Parcel In Transit');
            email.setPlainTextBody('Your parcel with ID ' + parcelId + ' is now in transit.');
        } else if (status == 'Out for Delivery') {
            email.setSubject('Parcel Out for Delivery');
            email.setPlainTextBody('Your parcel with ID ' + parcelId + ' is out for delivery and will be delivered soon.');
        } else if (status == 'Delivered') {
            email.setSubject('Parcel Delivered');
            email.setPlainTextBody('Your parcel with ID ' + parcelId + ' has been delivered.');
        } else if (status == 'On Hold') {
            email.setSubject('Parcel On Hold');
            email.setPlainTextBody('The delivery of your parcel with ID ' + parcelId + ' is temporarily on hold.');
        } else if (status == 'Failed Delivery Attempt') {
            email.setSubject('Parcel Delivery Attempt Failed');
            email.setPlainTextBody('An attempt to deliver your parcel with ID ' + parcelId + ' was unsuccessful.');
        } else if (status == 'Returned') {
            email.setSubject('Parcel Returned');
```

Apex Class :

```
public class ParcelNotificationClass {

    public static void sendNotification(String recipientEmail, String parcelId, String status)
{
```

```
// Compose the email message based on the parcel status
Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
email.setToAddresses(new String[]{recipientEmail});

if (status == 'Pickup') {
    email.setSubject('Parcel Scheduled for Pickup');
    email.setPlainTextBody('Your parcel with ID ' + parcelId + ' is scheduled for pickup.');
} else if (status == 'In Transit') {
    email.setSubject('Parcel In Transit');
    email.setPlainTextBody('Your parcel with ID ' + parcelId + ' is now in transit.');
} else if (status == 'Out for Delivery') {
    email.setSubject('Parcel Out for Delivery');
    email.setPlainTextBody('Your parcel with ID ' + parcelId + ' is out for delivery and will be delivered soon.');
} else if (status == 'Delivered') {
    email.setSubject('Parcel Delivered');
    email.setPlainTextBody('Your parcel with ID ' + parcelId + ' has been delivered.');
} else if (status == 'On Hold') {
    email.setSubject('Parcel On Hold');
    email.setPlainTextBody('The delivery of your parcel with ID ' + parcelId + ' is temporarily on hold.');
```

```
        } else if (status == 'Failed Delivery Attempt') {

            email.setSubject('Parcel Delivery Attempt Failed');

            email.setPlainTextBody('An attempt to deliver your parcel with ID ' + parcelId + '
was unsuccessful.');

        } else if (status == 'Returned') {

            email.setSubject('Parcel Returned');

            email.setPlainTextBody('Your parcel with ID ' + parcelId + ' has been returned to
the sender.');

        } else {

            // Handle other statuses or provide a default message if needed

            email.setSubject('Parcel Status Update');

            email.setPlainTextBody('Parcel ID ' + parcelId + ' has changed status to ' +
status);

        }



        // Send the email

        Messaging.sendEmail(new Messaging.SingleEmailMessage[]{email});

    }

}
```

You can schedule a job to execute this method.

```apex
public class ParcelNotificationScheduler implements Schedulable {
    public void execute(SchedulableContext sc) {
        // Call the method within ParcelNotificationClass
        ParcelNotificationClass.sendNotification('recipient@example.com', 'Parcel123', 'Delivered');
    }
}
```

Activity- 2 Use Case: This Trigger works to handle parcel updates and automate actions based on trigger
events, This apex trigger changes the status of parcels from "Pickup" to "In Transit" and sends a notification when this change occurs.

Trigger:

```apex
trigger ParcelStatusUpdate on Parcel__c (after update) {
    List<Parcel__c> parcelsToUpdate = new List<Parcel__c>();

    for (Parcel__c parcel : Trigger.new) {
        String oldStatus = Trigger.oldMap.get(parcel.Id).ParcelStatus__c;
        String newStatus = parcel.ParcelStatus__c;

        if (oldStatus != newStatus) {
            String notificationMessage = 'Parcel ID ' + parcel.Id + ' has changed status from ' + oldStatus + ' to ' + newStatus;

            if (newStatus == 'In Transit' || newStatus == 'Out for Delivery' || newStatus == 'Delivered' || newStatus == 'On Hold' || newStatus ==
                parcelsToUpdate.add(parcel);

                // Send a notification (you would implement this part according to your needs)
                // Example: Sending an email
                Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
                email.setToAddresses(new String[] { 'recipient@example.com' });
                email.setSubject('Parcel Status Update');
                email.setPlainTextBody(notificationMessage);
                Messaging.sendEmail(new Messaging.SingleEmailMessage[] { email });
            }
        }
    }

    // Update parcel status if needed
    if (!parcelsToUpdate.isEmpty()) {
```

Trigger Code:

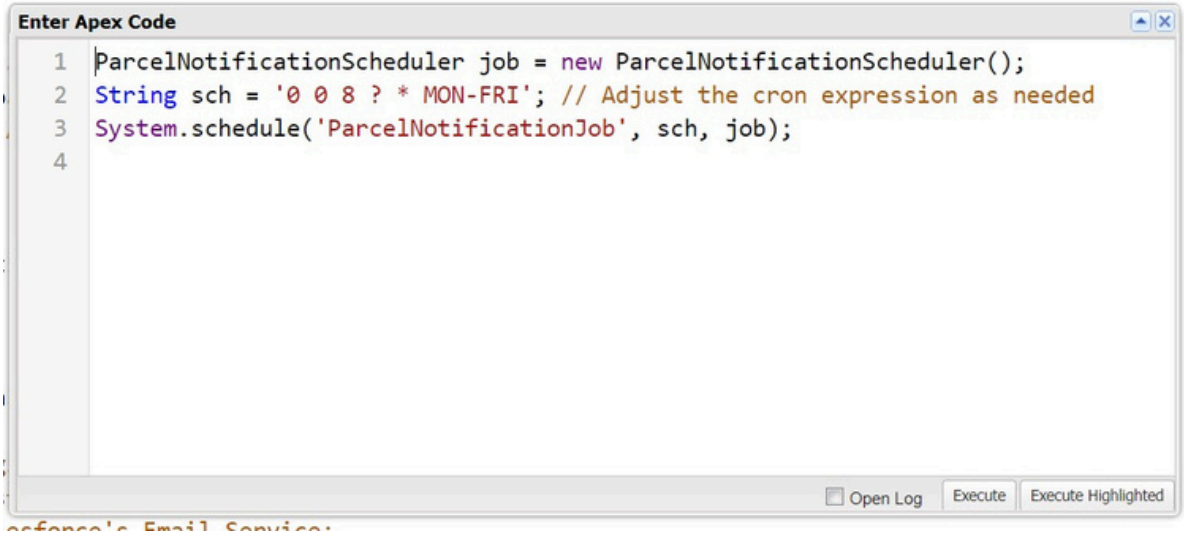
```
trigger ParcelStatusUpdate on Parcel__c (after update) {
   List<Parcel__c> parcelsToUpdate = new List<Parcel__c>();

   for (Parcel__c parcel : Trigger.new) {
      String oldStatus = Trigger.oldMap.get(parcel.Id).ParcelStatus__c;
      String newStatus = parcel.ParcelStatus__c;

      if (oldStatus != newStatus) {
         String notificationMessage = 'Parcel ID ' + parcel.Id + ' has changed status from ' + oldStatus +
' to ' + newStatus;

if (newStatus == 'In Transit' || newStatus == 'Out for Delivery' || newStatus == 'Delivered' || newStatus
== 'On Hold' || newStatus == 'Scheduled for Pickup' || newStatus == 'Failed Delivery Attempt' ||
newStatus == 'Returned') {
            parcelsToUpdate.add(parcel);

            // Send a notification (you would implement this part according to your needs)
            // Example: Sending an email
            Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
            email.setToAddresses(new String[] { 'recipient@example.com' });
            email.setSubject('Parcel Status Update');
            email.setPlainTextBody(notificationMessage);
            Messaging.sendEmail(new Messaging.SingleEmailMessage[] { email });
         }
      }
   }

   // Update parcel status if
   needed if
   (!parcelsToUpdate.isEmpty()) {
      for (Parcel__c updatedParcel : parcelsToUpdate) {
         updatedParcel.ParcelStatus__c = updatedParcel.ParcelStatus__c;
      }
      update parcelsToUpdate;
   }
}
```

Activity- 3

Apex Scheduler Class :

```
public class ParcelNotificationScheduler implements Schedulable {
   public void execute(SchedulableContext sc) {
      // Call the method within ParcelNotificationClass
      ParcelNotificationClass.sendNotification('recipient@example.com', 'Parcel123', 'Delivered');
   }
}
```

To schedule the job to run the method at specific times and frequencies, you need to use the Salesforce user interface or execute the scheduling code using anonymous Apex.




```
ParcelNotificationScheduler job = new ParcelNotificationScheduler();
String sch = '0 0 8 ? * MON-FRI'; // Adjust the cron expression as needed
System.schedule('ParcelNotificationJob', sch, job);
```

The above code schedules the "ParcelNotificationScheduler" class to run the execute method (which calls the sendNotification method in "ParcelNotificationClass") at 8 AM from Monday to Friday.

Milestone 11: Login Page Component (LWC)

Use Case upon successful login, the application displays a personalized dashboard with the following details for each parcel associated with the customer:

- ParcelID
- Status
- DeliveryDate
- DeliveryPersonnelName
- ContactNumber

Activity- 1

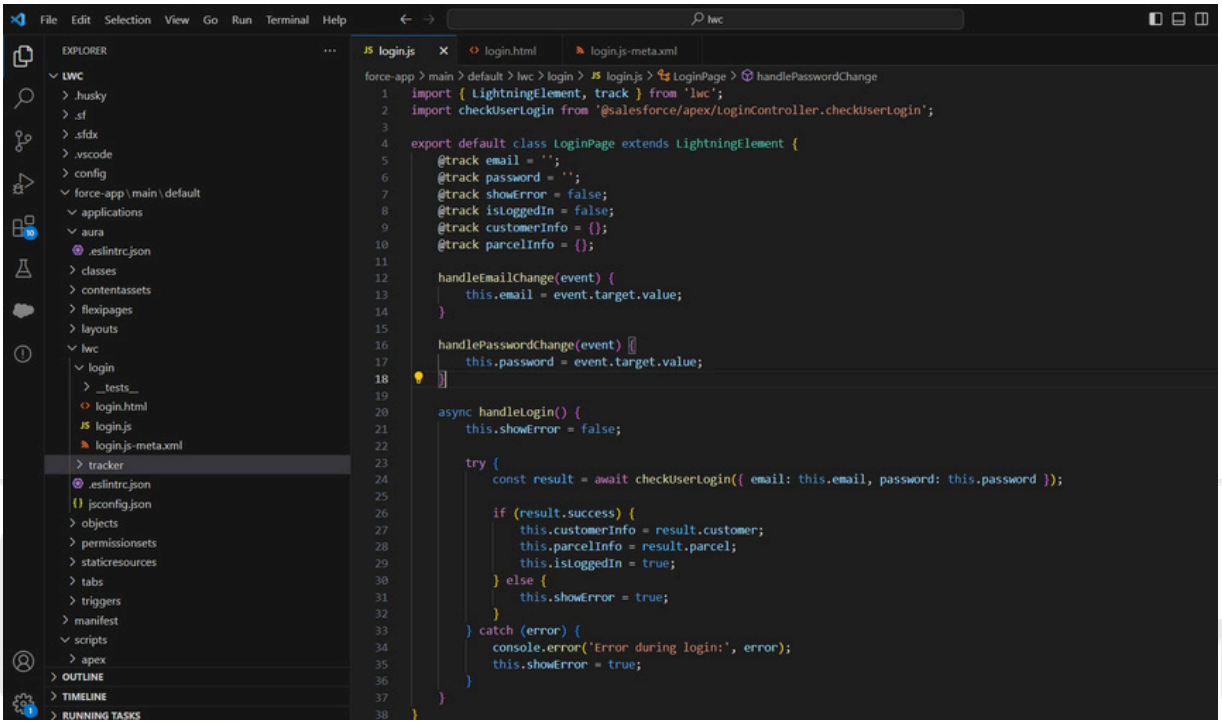
Open a Workspace:
Open Visual Studio Code and create a new workspace or open an existing one. A workspace is a directory that contains your Salesforce projects.

Create a New Salesforce Project:
Open the Command Palette (Ctrl + Shift + P) and run the command "SFDX: Create Project".
Choose the project type.
For LWC development, choose "Standard" for most cases.

Authorize an Org:
Open the Command Palette and run the command "SFDX: Authorize an Org".
Log in to your Salesforce org.

Create a New Lightning Web Component:
Open the Command Palette and run the command "SFDX: Create Lightning Web Component".
Enter a name for your component (Login) and choose the directory to save it.

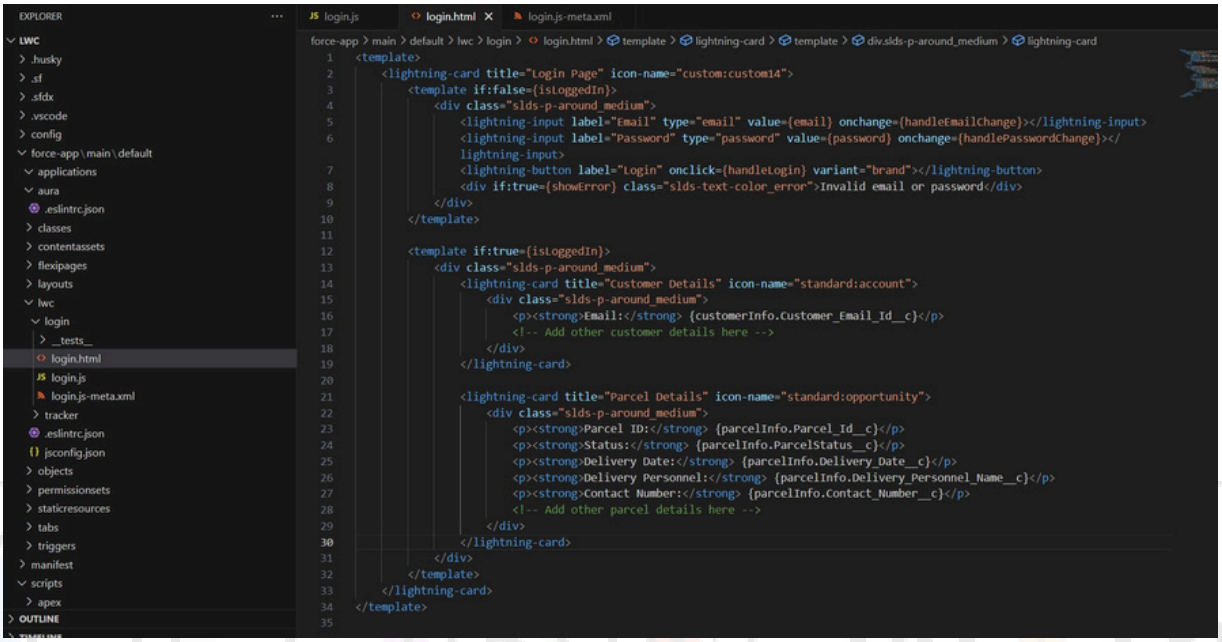
Edit Your Lightning Web Component:
Open the newly created component in the force-app/main/default/lwc directory.
Edit the HTML, JavaScript, and CSS files as needed.



Login (JavaScript) code:

```
import { LightningElement, track } from 'lwc';
import checkUserLogin from '@salesforce/apex/LoginController.checkUserLogin';

export default class LoginPage extends LightningElement {
  @track email = '';
  @track password = '';
  @track showError = false;
  @track isLoggedIn = false;
  @track customerInfo = {};
  @track parcelInfo = {};

  handleEmailChange(event) {
    this.email = event.target.value;
  }

  handlePasswordChange(event) {
```

```
        this.password = event.target.value;
    }

    async handleLogin() {
        this.showError = false;

        try{
            const result = await checkUserLogin({ email: this.email, password: this.password
});

            if (result.success) {
                this.customerInfo = result.customer;
                this.parcelInfo = result.parcel;
                this.isLoggedIn = true;
            } else {
                this.showError = true;
            }
        } catch (error) {
            console.error('Error during login:', error);
            this.showError = true;
        }
    }
}
```



Login (html) code:

```
<template>
  <lightning-card title="Login Page" icon-name="custom:custom14">
    <template if:false={isLoggedIn}>
      <div class="slds-p-around_medium">
        <lightning-input label="Email" type="email" value={email}
onchange={handleEmailChange}></lightning-input>
        <lightning-input label="Password" type="password" value={password}
onchange={handlePasswordChange}></lightning-input>
        <lightning-button label="Login" onclick={handleLogin}
variant="brand"></lightning-button>
        <div if:true={showError} class="slds-text-color_error">Invalid email or
password</div>
      </div>
    </template>

    <template if:true={isLoggedIn}>
      <div class="slds-p-around_medium">
        <lightning-card title="Customer Details" icon-name="standard:account">
```

Login (xml) code:

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
  <apiVersion>58.0</apiVersion>
  <isExposed>true</isExposed>
  <targets>
    <target>lightning__AppPage</target>
    <target>lightning__RecordPage</target>
    <target>lightning__HomePage</target>
  </targets>
</LightningComponentBundle>
```

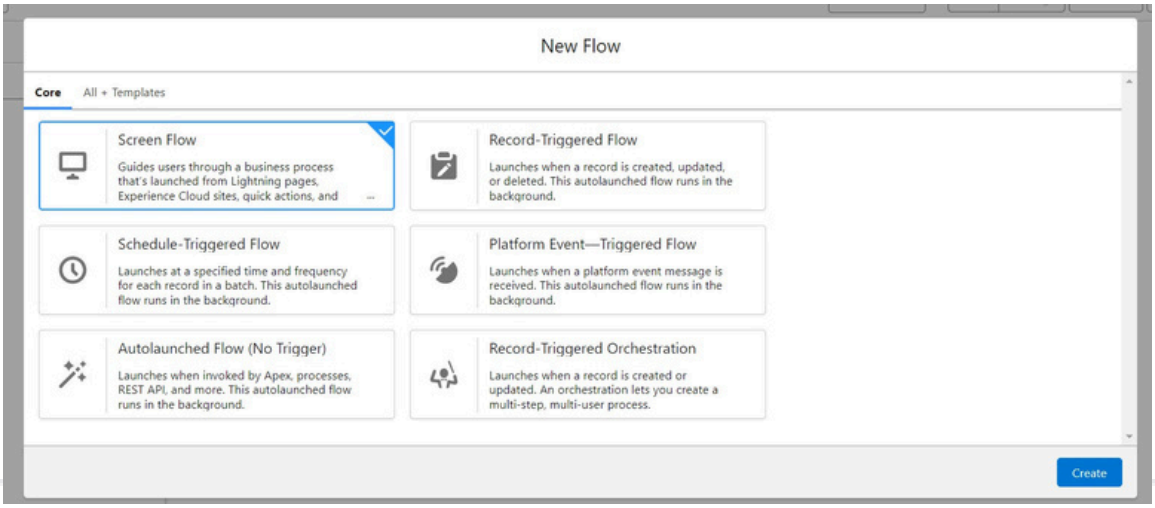
Activity- 2

Apex Controller (LoginController.cls):

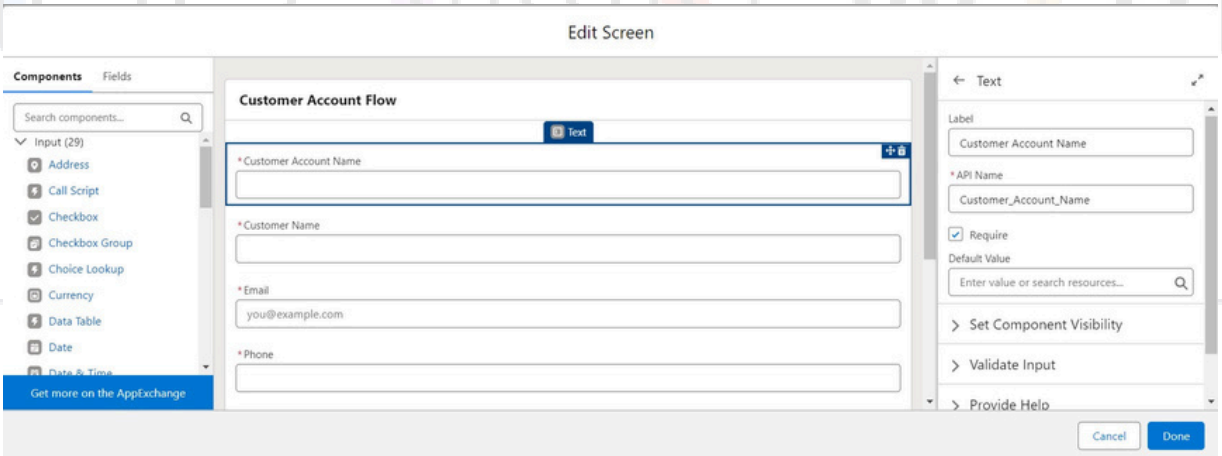
```
public with sharing class LoginController {
    @AuraEnabled(cacheable=true)
    public static Map<String, Object> checkUserLogin(String email, String password) {
        Map<String, Object> result = new Map<String, Object>();

        // Query the Customer_Account__c object to check if the email and password match
        List<Customer_Account__c> accounts = [SELECT Id, Customer_Email_Id__c FROM Customer_Account__c WHERE Customer_Email_Id__c = :email AND Pass

        if (!accounts.isEmpty()) {
            Customer_Account__c account = accounts[0];

            // You may retrieve additional customer information here if needed
            // Example: Map<String, Object> customerDetails = fetchAdditionalCustomerDetails(account.Id);

            // Return success and relevant information
            result.put('success', true);
            result.put('customer', account);
            result.put('parcel', fetchParcelDetails(account.Id)); // Assume fetchParcelDetails is another method to retrieve parcel details
        } else {
            result.put('success', false);
        }

        return result;
    }

    // Example method to retrieve parcel details based on customer ID
```

```
public with sharing class LoginController {
  @AuraEnabled(cacheable=true)
  public static Map<String, Object> checkUserLogin(String email, String password) {
    Map<String, Object> result = new Map<String, Object>();
```

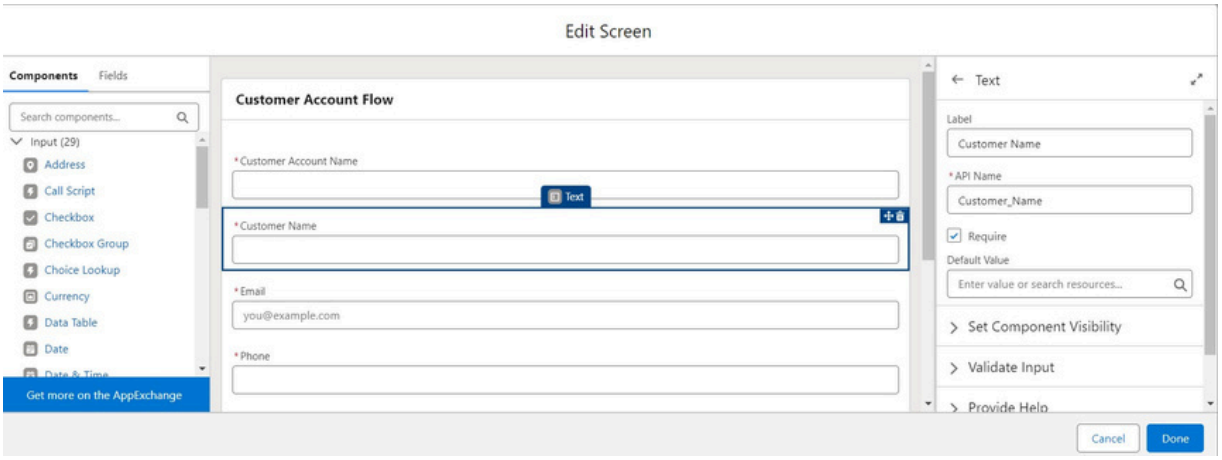
```
        // Query the Customer_Account__c object to check if the email and password
match
        List<Customer_Account__c> accounts = [SELECT Id, Customer_Email_Id__c FROM
Customer_Account__c WHERE Customer_Email_Id__c = :email AND Password__c =
:password LIMIT 1];

        if (!accounts.isEmpty()) {
            Customer_Account__c account = accounts[0];

            // You may retrieve additional customer information here if needed
            // Example: Map<String, Object> customerDetails =
fetchAdditionalCustomerDetails(account.Id);

            // Return success and relevant information
            result.put('success', true);
            result.put('customer', account);
            result.put('parcel', fetchParcelDetails(account.Id)); // Assume fetchParcelDetails
is another method to retrieve parcel details
        } else {
            result.put('success', false);
        }

        return result;
    }

    // Example method to retrieve parcel details based on customer ID
    private static Map<String, Object> fetchParcelDetails(Id customerId) {
        Map<String, Object> parcelDetails = new Map<String, Object>();

        // Implement your logic to query parcel details based on the customer ID
        // Example: Parcel__c parcel = [SELECT Id, Parcel_Details__c FROM Parcel__c
WHERE Customer__c = :customerId LIMIT 1];

        // Add relevant parcel information to the parcelDetails map
```



3. Under the Screen Flow Click on “+” Symbol and In the Drop down List select the “Screen Element”.

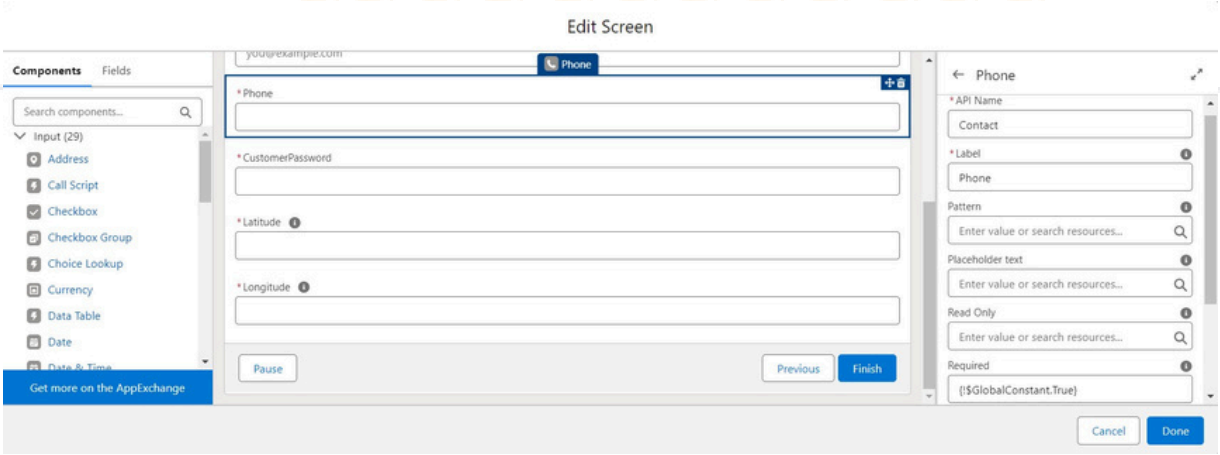


4. Drag the “Text” component in to the flow element → Label “Customer Account Name” → Require checkbox should be selected →Done.
5. Drag the “Text” component in to the flow element → Label “Customer Name” → Require checkbox should be selected →Done.



6. Drag the “Email” component in to the flow element → API Name “EmailId” Label “EmailId” should be selected from the drop down → Disabled “{!$GlobalConstant.False}” Read Only “{!$GlobalConstant.False}” → Required “{!$GlobalConstant.True}” →Done.

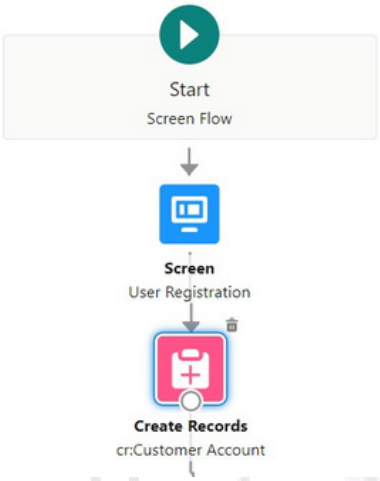
7. Drag the “Phone” component in to the flow element → API Name “Contact” Label “Phone” should be selected from the drop down → Required “{!$GlobalConstant.True}” →Done.



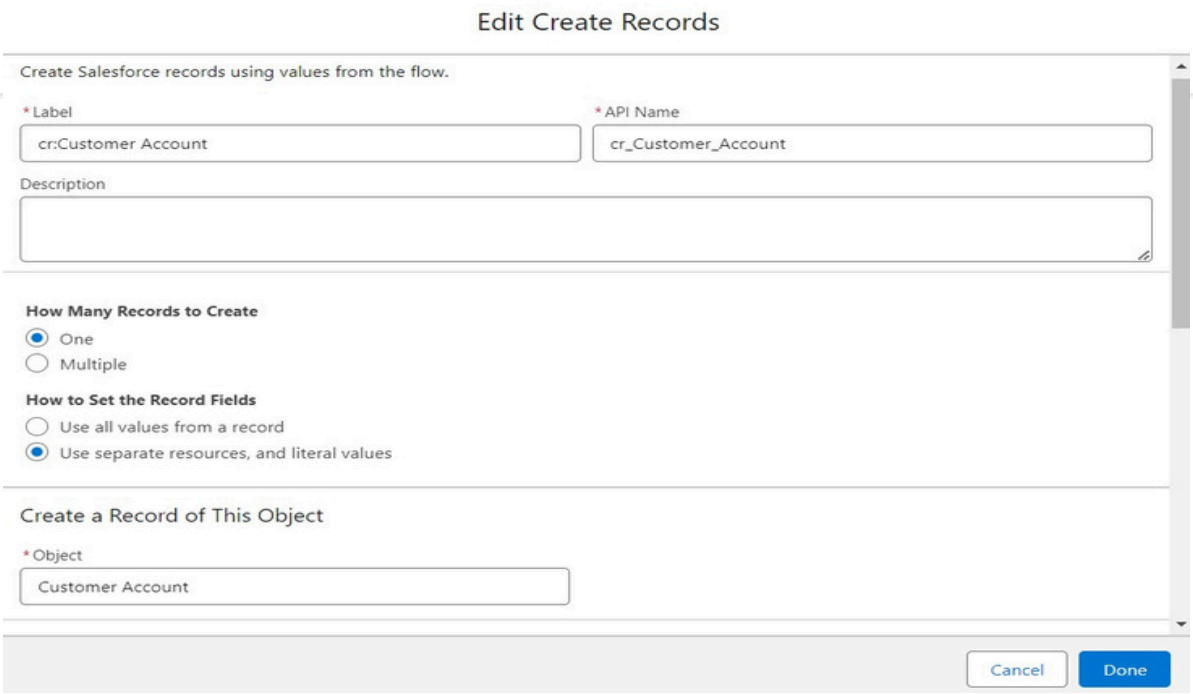
8. Drag the “Password” component in to the flow element → Label “Customer Password” → Required “{!$GlobalConstant.True}” →Done.

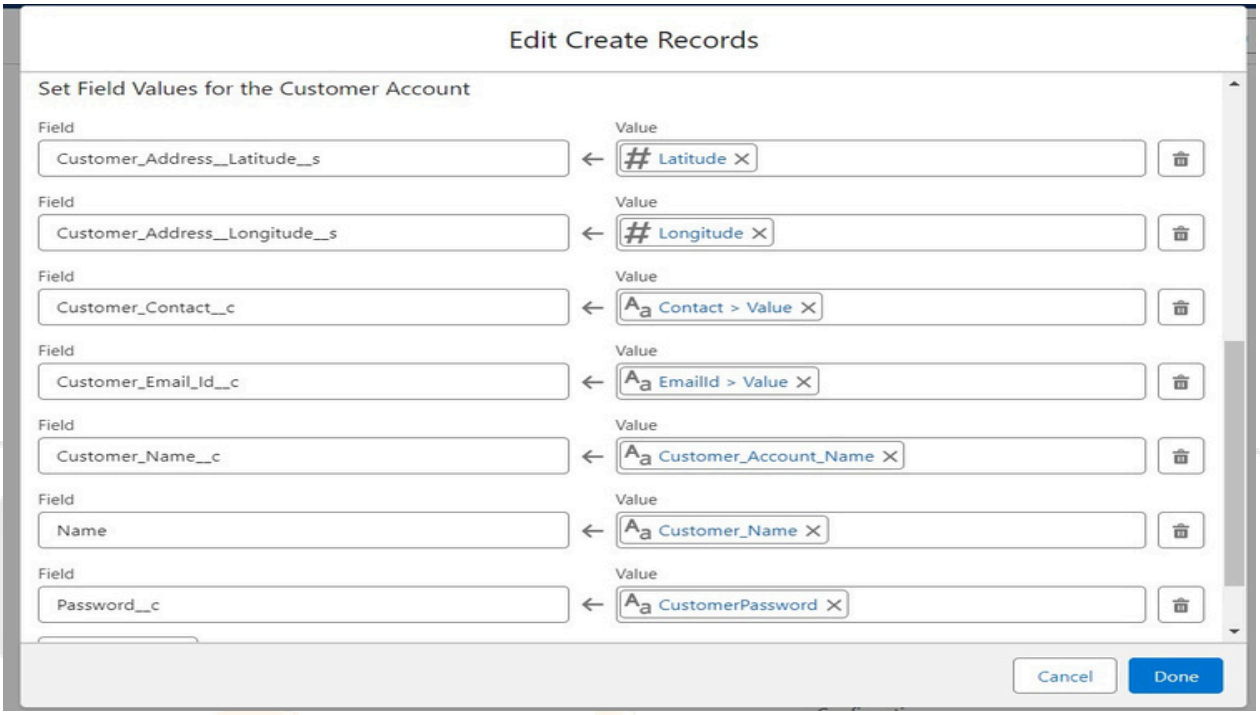
Activity 2:

1. Under the Screen Element Click on “+” Symbol and In the Drop down List select the “Create Records Element”.



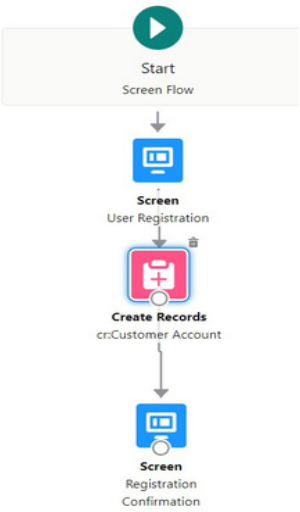
2. Under the Create a record for this object “Customer Account” → Set Fields for Customer Account → Click Add Field And add the respective fields as shown → Done.





Activity 3:

1. Under the Create Records Element Click on “+” Symbol and In the Drop down List select the “Screen Element”.



Project Related Questions:

1. Which Salesforce object is used to store information about the parcels, including Parcel ID, Sender Name, Receiver Name, Weight, Size, and Delivery Address?

a. Account
b. Contact
c. Custom Object - Parcel
d.Opportunity Solution:c

2. In the Parcel object, a formula field named "Delivery Time" is needed. What type of formula should be used to calculate the time taken for delivery based on the Pickup Date and Estimated Delivery Date?

a. Text
b. Number
c. Date/Time
d.Formulafieldscan'tperformcalculationsondates Solution:c

3. Which of the following is true about Apex in the Salesforce platform?

a. Apex is a declarative language.
b. Apex is used for designing page layouts.
c. Apex is executed on the client-side.
d. Apex is a server-side programming language for building business processes.
Solution: d

4.What is the main purpose of an Apex trigger in the context of the Parcel Dispatcher project?

a. To define custom fields on objects.
b. To automate actions based on events like changing the status of a parcel.
c. To create user interfaces for the project.
d.Togeneratereportsandanalytics. Solution:b

5.What is the primary advantage of using Lightning Web Components (LWC) in the Parcel Dispatcher project?

a. LWC provides declarative features for building forms.
b. LWC facilitates server-side processing.
c. LWC allows for building responsive and efficient user interfaces.
d.LWCisspecificallydesignedforwritingbackendlogic. Solution:c

6. What is the purpose of using Schedule Apex in the Parcel Dispatcher project?

a. To create schedules for parcel pickups.
b. To automate certain tasks at specific times, such as sending notifications to customers.
c. To define the delivery routes for delivery personnel.
d.Togeneratereportsonparceldispatchefficiency. Solution:b

7. Which type of Salesforce report would be most suitable for tracking the status of parcels based on different record types (Pickup, In Transit, Delivered)?

a. Tabular Report
b. Summary Report
c. Matrix Report
d.JoinedReport Solution:c

8.What role do Flows play in the Parcel Dispatcher project?

a. Flows are used to design user interfaces for the application.
b. Flows automate business processes and guide users through the steps.
c. Flows manage the database relationships between objects.
d.Flowsareusedfordefiningsecuritysettings. Solution:b

9. In the project, which Salesforce object is used to store information about delivery personnel, including Name, Email, Phone, and Route?

a. Account
b. Contact
c. Custom Object - Delivery Personnel
d.Opportunity Solution:c

10.Which of the following is true about formula fields in Salesforce?

a. Formula fields can only reference fields on the same object.
b. Formula fields can reference fields on related objects.
c. Formula fields can execute Apex code.
d.Formulafieldsareusedfordefiningrecordtypes. Solution:b

11.When does an Apex trigger execute in Salesforce?

a. During the data loading process.
b. After a record is inserted, updated, or deleted.
c. When a user logs in.
d. Apex triggers only run when explicitly invoked by the user. Solution: b

12.What is the key feature of Lightning Web Components (LWC) that enhances performance?

a. LWC uses server-side rendering.
b. LWC components are automatically cached.
c. LWC supports asynchronous loading of components.
d.LWCcomponentsarebuiltwithVisualforce. Solution:c

13.Which of the following statements about Schedule Apex is correct?

a. Schedule Apex is used for real-time event handling.
b. Schedule Apex can only be run once.

c. Schedule Apex is suitable for background jobs and periodic tasks.
d. Schedule Apex is designed for user interface development. Solution: c

14.What is the primary purpose of dashboards in Salesforce?

a. To define security settings.
b. To create workflows.
c. To visualize and analyze data from reports.
d.Todesignuserinterfaces. Solution:c

15. In the Parcel Dispatcher project, how can Flows be utilized for automation?

a. Flows can be used to design website layouts.
b. Flows can automate repetitive tasks and guide users through complex processes.
c. Flows are used to define relationships between objects.
d.Flowscangenerateescheduledreports. Solution:b