

Boutique App CRM

The Boutique Management App is a custom Salesforce application designed to help boutique owners seamlessly manage their customers, products, and sales in one centralized platform.

Problem Statement

Boutiques often face challenges in managing inventory, handling customer appointments, and delivering personalized experiences due to reliance on manual processes or basic systems. This leads to issues like overlapping appointments, stock mismatches, and limited visibility into sales trends. To overcome these challenges, a custom Boutique Management App is needed to centralize operations, prevent errors, streamline appointment scheduling, track products efficiently, and provide managers with insightful reports for better decision-making and improved customer satisfaction.

Phase 1: Problem Understanding & Industry Analysis

1. Requirement Gathering

Goal: Understand what each stakeholder needs from the boutique management system.

- **Customers:** Easy browsing of collections, personalized recommendations, loyalty rewards, seamless online & in-store experience.
- **Business Owners:** End-to-end visibility on sales, inventory, customer engagement, and profitability.

2. Stakeholder Analysis

Goal: Identify roles and responsibilities in the ecosystem.

- **Primary Stakeholders:**
 - **Customers** → End users, purchasing products online or in-store.
 - **Sales Staff** → Assist with transactions, update customer preferences, and upsell products.
 - **Store Manager** → Manages daily operations, inventory, and staff schedules.
- **Secondary Stakeholders:**
 - **Business Owner/Management** → Strategic decisions on sales, marketing, and expansion.
 - **IT/Salesforce Admins** → Maintain the boutique CRM & integrations.
 - **Suppliers/Vendors** → Provide stock and manage restocking cycles.

3. Business Process Mapping

Goal: Understand current challenges vs. how Salesforce can improve operations.

Current Process (Manual/Traditional):

- Customer info tracked manually in diaries/spreadsheets.
- Inventory updated at end of the day → stock mismatches.
- Customer preferences rarely tracked → missed upsell opportunities.
- Marketing mostly word-of-mouth or generic promotions.

Proposed Process (Salesforce Enabled):

- Customer interactions captured automatically (POS, website, social media).

- Real-time inventory sync with alerts for low stock.
- Automated loyalty & rewards programs.
- Personalized fashion recommendations powered by Salesforce AI.
- Targeted marketing campaigns (SMS, email, WhatsApp) from Salesforce Marketing Cloud.

4. Industry-Specific Use Case Analysis

Goal: Benchmark against best practices in Boutique/Retail.

- **Customer Retention:**
 - Many boutiques lose customers due to lack of personalized follow-up.
 - **Solution** → Salesforce CRM with customer history, birthdays, style preferences, automated reminders.
- **Inventory Management:**
 - Stockouts or overstocking are common.
 - **Solution** → Inventory dashboard with predictive restocking.

5. AppExchange Exploration

Goal: Identify existing Salesforce apps to reduce development effort.

- **Potential Apps for Boutiques:**
 - POS/Inventory Management Apps.
 - Loyalty Management Apps (customer rewards & referrals).
 - Marketing Automation Apps (SMS, WhatsApp, Email campaigns).

Phase 2: Org Setup & Configuration (Boutique App)

Goal: Prepare Salesforce environment for the Boutique application.

1. Salesforce Editions

For the Boutique App, we will use a Salesforce Developer Edition Org (free) to build and test all features such as appointment scheduling, customer management, and order tracking. This Developer Org acts as a safe environment where we can configure, customize, and test without affecting real data. Once everything is finalized and tested, the solution can later be moved to a Production Org, where it will be used by the boutique staff to manage real customers, appointments, and sales transactions.

2. Company Profile Setup

For the Boutique App, I set up the company details with the boutique's name, address, and contact information, configured the time zone to match the boutique's location, and set the default currency to INR (₹) or USD (\$) depending on customer needs.

If you enter blank business hours for a day, that means your organization does not operate on that day.

Business Hours Edit

SaveCancel

Step 1. Business Hours Name

Required Information

Business Hours Name

Default

Use these business hours as the default

Active

Step 2. Time Zone

Time Zone

(GMT+05:30) India Standard Time (Asia/Kolkata)

Step 3. Business Hours

Sunday	9:00 AM	to	10:00 PM	<input type="checkbox"/> 24 hours
Monday	9:00 AM	to	10:00 PM	<input type="checkbox"/> 24 hours
Tuesday	9:00 AM	to	10:00 PM	<input type="checkbox"/> 24 hours
Wednesday	9:00 AM	to	10:00 PM	<input type="checkbox"/> 24 hours
Thursday	9:00 AM	to	10:00 PM	<input type="checkbox"/> 24 hours
Friday	9:00 AM	to	10:00 PM	<input type="checkbox"/> 24 hours
Saturday	9:00 AM	to	10:00 PM	<input type="checkbox"/> 24 hours

Save

Cancel

SETUP

Company Information

Locale Settings

Default Locale

English (United States)

Default Language

English

Default Time Zone

(GMT+05:30) India Standard Time (Asia/Kolkata)

Currency Settings

Currency Locale

Malayalam (India) - INR

Turning on multiple currencies introduces permanent changes in your org.

This feature can't be turned off. Review the [Implications of Enabling Multiple Currencies](#) before enabling.

Activate Multiple Currencies

☐

Translation Settings

Enable Data Translation

☐

Salesforce Newsletter Settings

☒ Users receive the Salesforce newsletter

☒ Users receive the Salesforce admin newsletter

Login Notifications

Hide Notices About System Maintenance

☐

Hide Notices About System Downtime

☐

3. Business Hours & Holidays

For the Boutique App, the business hours are set from 9:00 AM to 10:00 PM, aligning with typical boutique operating times. In addition, public holidays such as Diwali, Christmas, and other major festive days are marked as non-working days to ensure that customers cannot schedule appointments on those dates.

4. Fiscal Year Settings

The Standard Fiscal Year (Jan–Dec) is used for the revenue & sales reporting.

Fiscal Year Information

Your organization can change the fiscal year start month, and specify whether the fiscal year name is set to the starting or ending year. For example, if your fiscal year starts in April 2025 and ends in March 2026, your Fiscal Year setting can be either 2025 or 2026.

Changing the fiscal year shifts fiscal periods and impacts opportunities and forecasts across your organization. If your forecast periods are set to quarterly, adjusting the fiscal year start month will erase existing forecast adjustments and quotas. Consider exporting a data backup before implementing this change.

☒ Standard Fiscal Year

☐ Custom Fiscal Year

Change Fiscal Year Period

Save

Cancel

Name

Gyan Ganga Institute of Technology and Sciences

Fiscal Year Start Month

January

Fiscal Year is Based On

☒ The ending month
 ☐ The starting month

Save

Cancel

5. User Setup & Licenses

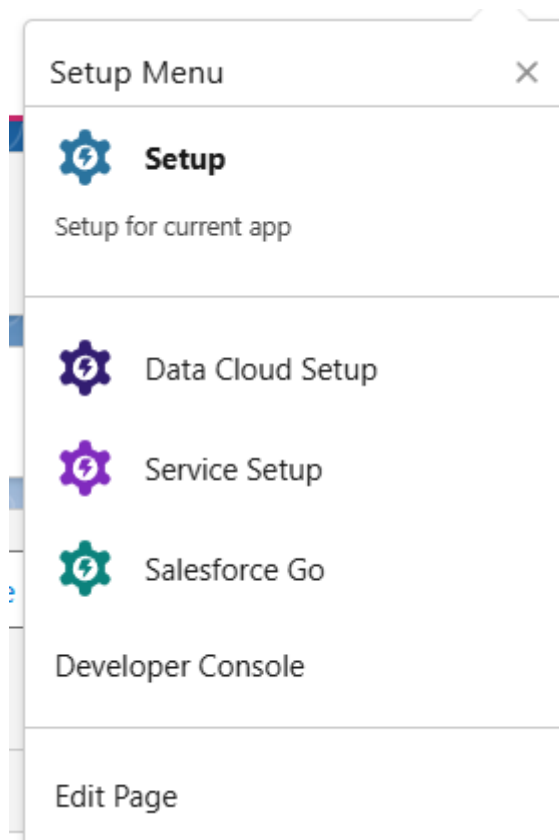
Since it is an application to manage a small-scale business so not much profile setup and licences is need.

6. Login Access Policies

For the Boutique App, staff login access is restricted to 10:00 AM – 8:00 PM, ensuring that employees can only log in during active working

7. Dev Org Setup

For the Boutique App, the Developer Org serves as a sandbox environment where all configurations, automation, Lightning components, and Apex development can be safely built and tested without affecting real customer data.



8. Deployment Basics

For the Boutique App, deployment involves moving all configurations and code—including custom objects, flows, Apex classes, and Lightning Web Components—from the sandbox environment to the production org. This can be done using tools such as Change Sets, VS Code with Salesforce CLI, or DevOps Center to ensure a smooth and controlled transition to the live environment.

Phase 3: Data Modelling & Relationships

Goal: Build data structure.

1. Standard & Custom Objects

- Standard: Contact (customers), Products, Orders.
- Custom: Alterations & Custom Orders, Appointments, Style Preferences, Supplier.

2. Fields

- Alteration & Custom Orders: Alterations & Custom Order Name, Delivery Date, Notes, Customer Name, Order Number
- Appointments: Appointment Name, Appointment Date, Purpose, Status
- Style Preferences: Style Preferences Name, Category, Colour, Size, Style Notes
- Supplier: Supplier Name, Contact Number, Product

Screenshots

The screenshot shows the Salesforce Setup interface, specifically the 'Object Manager' section for 'Alterations & Custom Order'. The left sidebar lists various setup options, with 'Fields & Relationships' selected. The main content area displays a table of fields for this object, sorted by field label. The table includes columns for Field Label, Field Name, Data Type, Controlling Field, and Indexed status. The fields listed are: Alterations & Custom Order Name (Text(80)), Created By (Lookup(User)), Customer Name (Lookup(Contact)), Delivery Date (Date), Last Modified By (Lookup(User)), Notes (Text Area(255)), Order Number (Auto Number), and Owner (Lookup(User,Group)).

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Alterations & Custom Order Name	Name	Text(80)		✓
Created By	CreatedById	Lookup(User)		
Customer Name	Customer_Name__c	Lookup(Contact)		✓
Delivery Date	Delivery_Date__c	Date		
Last Modified By	LastModifiedById	Lookup(User)		
Notes	Notes__c	Text Area(255)		
Order Number	Order_Number__c	Auto Number		
Owner	OwnerId	Lookup(User,Group)		✓



Search Setup



Setup Home Object Manager

Appointment

Details

Fields & Relationships

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

Object Limits

Record Types

Related Lookup Filters

Restriction Rules

Scoping Rules

Object Access

Triggers

Fields & Relationships

7 Items, Sorted by Field Label

Quick Find

New

Deleted Fields

Field Dependencies

Set History Tracking

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Appointment Date	Appointment_Date__c	Date		
Appointment Name	Name	Text(80)		✓
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Purpose	Purpose__c	Text Area(255)		
Status	Status__c	Picklist		



Search Setup



Setup Home Object Manager

Supplier

Details

Fields & Relationships

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

Object Limits

Record Types

Related Lookup Filters

Restriction Rules

Scoping Rules

Object Access

Triggers

Fields & Relationships

6 Items, Sorted by Field Label

Quick Find

New

Deleted Fields

Field Dependencies

Set History Tracking

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Contact Number	Contact_Number__c	Phone		
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Product	Product__c	Lookup(Product)		✓
Supplier Name	Name	Text(80)		✓



Search Setup



Setup Home Object Manager

Style Preference

Details

Fields & Relationships

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

Object Limits

Record Types

Related Lookup Filters

Restriction Rules

Scoping Rules

Object Access

Triggers

Fields & Relationships

8 Items, Sorted by Field Label

Quick Find

New

Deleted Fields

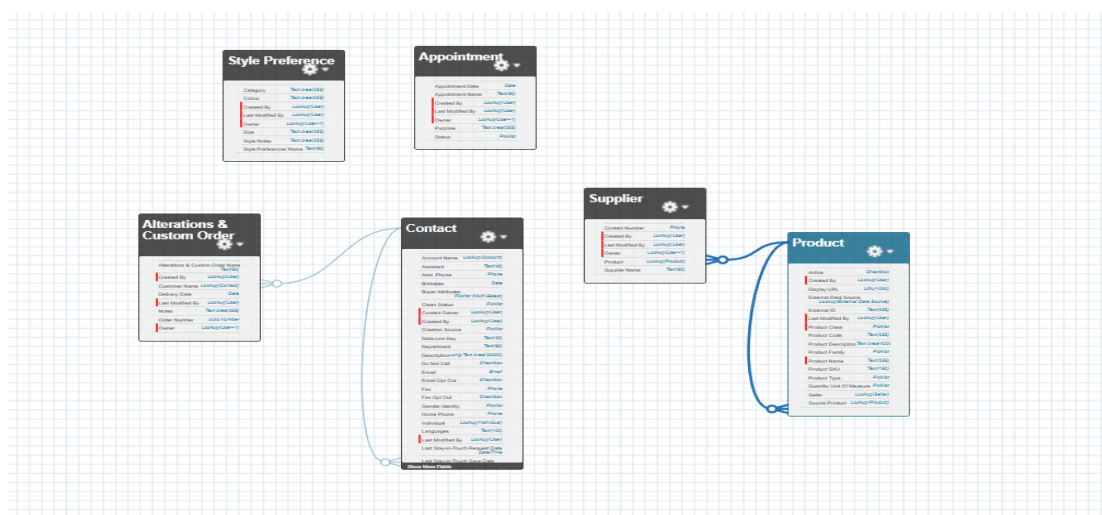
Field Dependencies

Set History Tracking

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Category	Category__c	Text Area(255)		
Colour	Colour__c	Text Area(255)		
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Size	Size__c	Text Area(255)		
Style Notes	Style_Notes__c	Text Area(255)		
Style Preferences Name	Name	Text(80)		✓

3. Schema Builder

Schema Builder serves as a visual blueprint of how the boutique's data is structured and connected. It allows you to easily create and manage custom objects like Customers, Appointments, and Products, while also defining the relationships between them, such as linking customers to their appointments or associating products with those appointments. Instead of just looking at lists of fields, Schema Builder presents everything in a clear diagram, making it easier to understand how data flows across the system. For the boutique, this means being able to track how customers schedule appointments, what services or products are included, and how these records interact, ensuring smooth operations and avoiding gaps or duplication in the system.



Phase 4: Process Automation (Admin)

Goal: Automate tasks.

1. Validation Rules

- Appointment Date must be not from past.
- Delivery Date must be not from past.

▼

+

🔍

?

⚙️

🔔

👤

Setup

Home

Object Manager ▼

SETUP

Object Manager

Alterations & Custom Order Validation Rule

Help for this Page

Back to Alterations & Custom Order

Validation Rule Detail

Edit

Clone

Rule Name	Delivery_Date_In_Future	Active	✓
Error Condition Formula	Delivery_Date__c < TODAY()		
Error Message	Delivery Date must be today or a future date.	Error Location	Delivery Date
Description			
Created By	Jaya Kushwaha 9/23/2025, 6:29 AM	Modified By	Jaya Kushwaha 9/23/2025, 6:29 AM

EditClone

▼

+

🔍

?

⚙️

🔔

👤

Setup

Home

Object Manager ▼

SETUP

Object Manager

Appointment Validation Rule

Help for this Page

Back to Appointment

Validation Rule Detail

Edit

Clone

Rule Name	Appointment_Date_In_Future	Active	✓
Error Condition Formula	Appointment_Date__c < TODAY()		
Error Message	Appointment Date must be today or a future date.	Error Location	Appointment Date
Description			
Created By	Jaya Kushwaha 9/23/2025, 6:32 AM	Modified By	Jaya Kushwaha 9/23/2025, 6:32 AM

EditClone

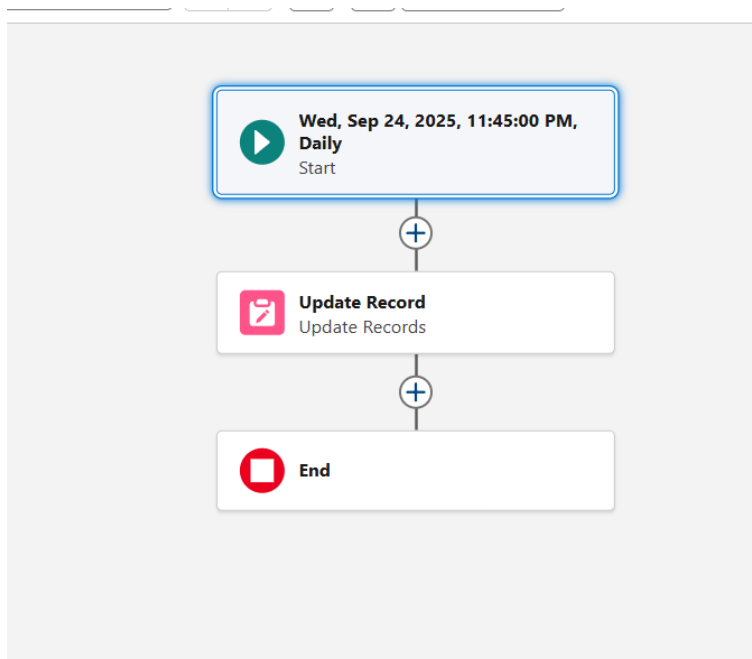
2. Email Alerts

- Customer gets email after approval.

The screenshot shows the 'Appointment Confirmation' email template in the 'Boutique App'. The interface includes a top navigation bar with a search bar and various menu items. The main content area is divided into 'Details' and 'Related' tabs. The 'Details' tab is active, showing fields for 'Email Template Name', 'Appointment Confirmation', 'Description', 'Made in Email Template Builder', 'Related Entity Type', 'Appointment Category', 'Folder', and 'Public Email Templates'. The 'Message Content' section displays the email body text: 'Your Appointment is Confirmed', 'Hello,', 'Your appointment has been successfully booked.', 'We look forward to seeing you!', 'Best regards,', and 'Team'.

3. Field Updates

- After the Appointment is completed the Status is updated to "Completed".



Phase 5: Apex Programming (Developer)

Goal: Add advanced logic.

1. Classes & Objects

Service Class: AppointmentService.cls

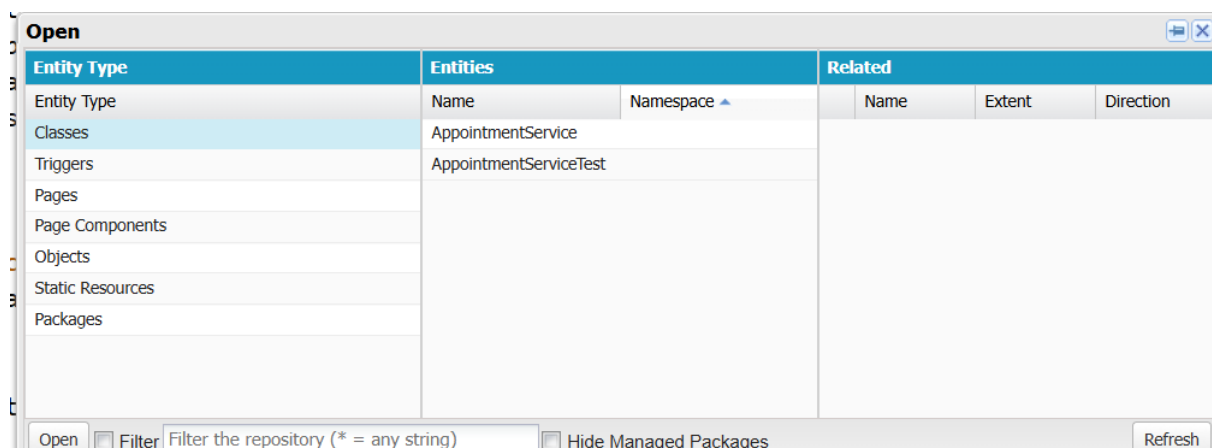
- This is a **helper class** that holds all the business logic for appointments.
It makes your code **reusable** and keeps your trigger clean.
- **What it does:**
 1. **scheduleAppointment()** → Sets the status to "Scheduled" (if date is today or future).
 2. **completeAppointment()** → Marks an appointment "Completed" only if the date is today or past.
 3. **cancelAppointment()** → Marks an appointment "Cancelled".

Test Class: AppointmentServiceTest.cls

- This is used to **prove that your code works** and to get Salesforce's required **75%+ coverage**.
- **What it does:**
 1. **Creates test Contacts (customers)** → since Appointment__c links to Customer__c.
 2. **Creates test Appointments** → scheduled for today or future.

3. Tests:

- Scheduling an appointment (should be "Scheduled").
- Completing an appointment (status changes to "Completed").
- Cancelling an appointment (status changes to "Cancelled").



2. Apex Triggers

Trigger: AppointmentTrigger

- This runs automatically when **Appointment__c** records are inserted or updated.
- **What it does:**
 1. **On Insert (new appointment):**
Calls `scheduleAppointment()` → ensures new appointments always start with correct status.
 2. **On Update:**
 - If status changed to "Completed" → calls `completeAppointment()`.

- If status changed to "Cancelled" → calls cancelAppointment().

Open

Entity Type	Entities		Related		
Entity Type	Name	Namespace	Name	Extent	Direction
Classes	AppointmentTrigger				
Triggers					
Pages					
Page Components					
Objects					
Static Resources					
Packages					

Open

☐ Filter

Filter the repository (* = any string)

☐ Hide Managed Packages

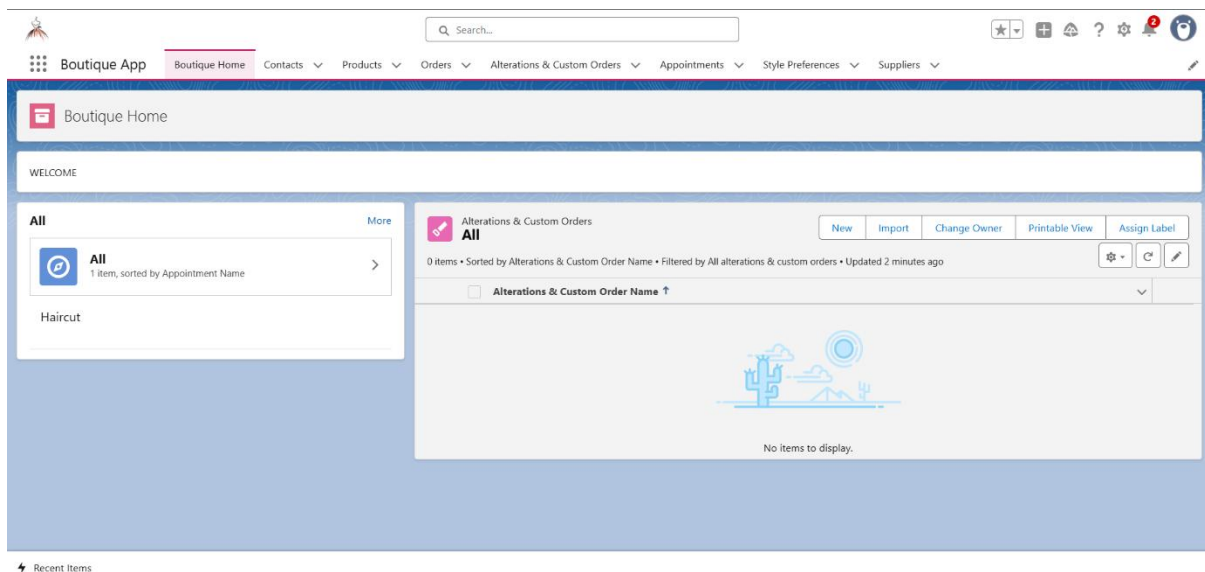
Refresh

Phase 6: User Interface Development

Goal: Make it user-friendly.

1. Lightning App Builder

- A boutique App is created with the Lightning Web Component for the handling of various tasks and customers in a small business.



2. Tabs

- Various tabs for the custom objects are added in the app for seamless use by the owner.

Custom Object Tabs			New	What Is This?
Action	Label	Tab Style	Description	
Edit Del	Alterations & Custom Orders	Handsaw		
Edit Del	Appointments	Compass		
Edit Del	Style Preferences	Cup		
Edit Del	Suppliers	Sailboat		

3. Apex with LWC

- Apex is used with lightning web component in integration with vs code for the better use of the application.

Open

Entity Type	Entities		Related		
Entity Type	Name	Namespace ▲	Name	Extent	Direction
Classes	AppointmentService				
Triggers	AppointmentServiceTest				
Pages					
Page Components					
Objects					
Static Resources					
Packages					

Open

☐ Filter

Filter the repository (* = any string)

☐ Hide Managed Packages

Refresh

Open

Entity Type	Entities		Related		
Entity Type	Name	Namespace	Name	Extent	Direction
Classes	AppointmentTrigger				
Triggers					
Pages					
Page Components					
Objects					
Static Resources					
Packages					

Open

☐ Filter

Filter the repository (* = any string)

☐ Hide Managed Packages

Refresh

```

force-app > main > default > lwc > upcomingAppointments > JS upcomingAppointments.js > ...
1  import { LightningElement, wire } from 'lwc';
2  import getTodayAppointments from '@salesforce/apex/AppointmentController.getTodayAppointments';
3
4  export default class UpcomingAppointments extends LightningElement {
5      @wire(getTodayAppointments) appointments;
6  }
7  |

```

```

force-app > main > default > lwc > upcomingAppointments > upcomingAppointments.js-meta.xml > ...
2  <LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
5      <targets>
6          <target>lightning__AppPage</target>
7          <target>lightning__HomePage</target>
8          <target>lightning__RecordPage</target>
9      </targets>
10 </LightningComponentBundle>
11

```

force-app > main > default > lwc > upcomingAppointments > <> upcomingAppointments.html > ...

```
1  <template>
2    <lightning-card title="Today's Appointments" icon-name="standard:contact">
3      <template if:true={appointments.data}>
4        <template for:each={appointments.data} for:item="app">
5          <div key={app.Id} class="slds-box slds-m-bottom_small slds-theme_default">
6            <p><strong>Customer:</strong> {app.Customer__r.Name}</p>
7            <p><strong>Date:</strong> {app.Appointment_Date__c}</p>
8            <p><strong>Status:</strong> {app.Status__c}</p>
9            <p><strong>Purpose:</strong> {app.Purpose__c}</p>
10         </div>
11       </template>
12     </template>
13     <template if:true={appointments.error}>
14       <p class="slds-text-color_error">Error loading appointments.</p>
15     </template>
16     <template if:false={appointments.data}>
17       <p>No appointments for today.</p>
18     </template>
19   </lightning-card>
20 </template>
21
```

force-app > main > default > classes > AppointmentService.cls > AppointmentService > scheduleAppointment(Appointment__c) : void

```
1  public with sharing class AppointmentService {
2
3    // Schedule a new appointment (before insert)
4    public static void scheduleAppointment(Appointment__c app) {
5      if (app.Appointment_Date__c == null) {
6        throw new AuraHandledException('Appointment Date is required.');
```

```
7      }
8      if (app.Appointment_Date__c < Date.today()) {
9        throw new AuraHandledException('Appointment Date must be today or in the future.');
```

```
10     }
11     app.Status__c = 'Scheduled';
12   }
13
14   // Mark as completed (before update when status changes)
15   public static void completeAppointment(Appointment__c app) {
16     if (app.Appointment_Date__c == null) {
17       throw new AuraHandledException('Appointment Date is required to complete an appointment.');
```

```
18     }
19     if (app.Appointment_Date__c <= Date.today()) {
20       app.Status__c = 'Completed';
21     } else {
22       throw new AuraHandledException('Appointment cannot be completed before the scheduled date.');
```

```
23     }
24   }
25
26   // Cancel an appointment
27   public static void cancelAppointment(Appointment__c app) {
28     app.Status__c = 'Cancelled';
29   }
30 }
31
```

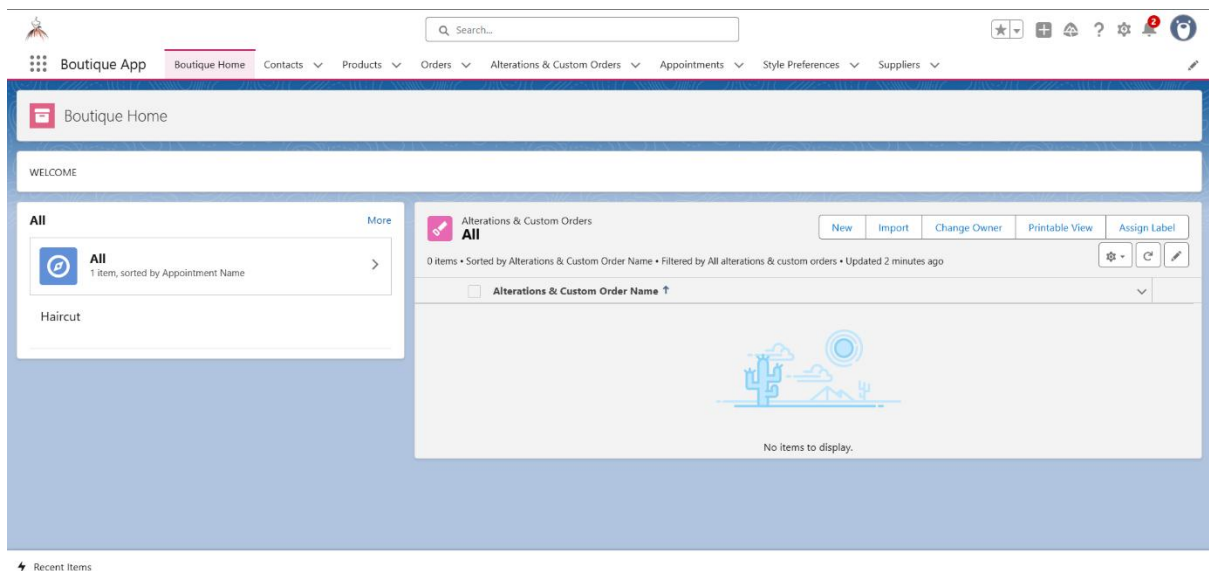
```
AppointmentService.cls  AppointmentTrigger.trigger  JS upcomingAppointments.js  upcomingAppointments.js-meta.xml

force-app > main > default > triggers > AppointmentTrigger.trigger > AppointmentTrigger

1  trigger AppointmentTrigger on Appointment__c (before insert, before update) {
2
3      if (Trigger.isBefore) {
4          if (Trigger.isInsert) {
5              for (Appointment__c app : Trigger.new) {
6                  AppointmentService.scheduleAppointment(app);
7              }
8          }
9
10         if (Trigger.isUpdate) {
11             for (Appointment__c app : Trigger.new) {
12                 Appointment__c oldApp = Trigger.oldMap.get(app.Id);
13
14                 if (app.Status__c == 'Completed' && oldApp.Status__c != 'Completed') {
15                     AppointmentService.completeAppointment(app);
16                 }
17
18                 if (app.Status__c == 'Cancelled' && oldApp.Status__c != 'Cancelled') {
19                     AppointmentService.cancelAppointment(app);
20                 }
21             }
22         }
23     }
24 }
```

4. Utility Bar

- Quick 'Recent Items' action for easy access by the owner.



Phase 7: Integration & External Access

Goal: Connect with outside systems.

1. Named Credentials

- Store insurance API credentials securely.

2. External Services

- Connect Salesforce to external systems like **email services** using **API**. Also lets us use feature like email when an appointment is created. In this, I've use REST API to notify the customer whenever the appointment is created.

The screenshot displays the Salesforce IDE interface. The top section shows the Apex code for `AppointmentTrigger2` and `BoutiqueWebService`. The bottom section shows a logs table with columns: User, Application, Operation, Time, Status, Read, and Size.

```
1 trigger AppointmentTrigger2 on Appointment__c (after insert) {
2   for(Appointment__c app : Trigger.new) {
3       BoutiqueWebService.sendAppointmentConfirmation(app.Id);
4   }
5 }
6
```

```
1 public with sharing class BoutiqueWebService {
2   @future(callout=true)
3   public static void sendAppointmentConfirmation(Id appointmentId) {
4       Appointment__c app = [SELECT Id, Customer__r.Email, Appointment_Date__c FROM Appointment__c WHERE Id = :appointmentId];
5
6       HttpRequest req = new HttpRequest();
7       req.setEndpoint('callout:Boutique_Email_API/send'); // Named Credential
8       req.setMethod('POST');
9       req.setHeader('Content-Type', 'application/json');
10      req.setBody(JSON.serialize(
11          new Map<String, Object>{
12              'email' => app.Customer__r.Email,
13              'date' => app.Appointment_Date__c,
14              'message' => 'Your appointment is confirmed!'
15          }
16      ));
17
18      Http http = new Http();
19      HttpResponse res = http.send(req);
20  }
21 }
22
```

User	Application	Operation	Time	Status	Read	Size
Jaya Kushwaha	Browser	/aura	9/25/2025, 2:50:10 PM	Success	Unread	784 bytes
Jaya Kushwaha	Browser	/aura	9/25/2025, 2:50:00 PM	Success	Unread	788 bytes
Jaya Kushwaha	Browser	/aura	9/25/2025, 2:47:58 PM	Success	Unread	743 KB
Jaya Kushwaha	Unknown	FutureHandler	9/25/2025, 2:47:58 PM	The callout couldn't access the endpoi...	Unread	6.42 KB
Jaya Kushwaha	Browser	/aura	9/25/2025, 2:45:21 PM	Success	Unread	1.98 KB

Phase 8: Data Management & Deployment

Goal: Manage data and move changes.

1. Data Import Wizard

- Import 50 demo Customer records from the Data Import Wizard.

Job ID	750gL00000E6NIB	Job Type	Bulk V1	Status	Closed
Submitted By	Jaya Kushwaha	Operation	Insert	Total Processing Time (ms)	0
Start Time	9/25/2025, 6:47 PM IST	Queued Batches	1	API Active Processing Time (ms)	0
End Time		In Progress Batches	0	Apex Processing Time (ms)	0
Time to Complete ([hh]:mm:ss)		Completed Batches	0		
Object	Contact	Failed Batches	0		
External ID Field		Progress	0%		
Content Type	CSV	Records Processed	0		
Concurrency Mode	Parallel	Records Failed	0		
API Version	64.0	Retries	0		
<div>AbortReload</div>					

Batches

View Request	View Result	Batch ID	Start Time	End Time	Total Processing Time (ms)	API Active Processing Time (ms)	Apex Processing Time (ms)	Records Processed	Records Failed	Retry Count	State Message	Status
View Request		751gL00000BfDmG	9/25/2025, 6:47 PM		0	0	0	0	0	0		Queued

2. Data Loader

- Import bulk Appointments from the Data Import Wizard.

Job ID	750gL00000E6c49	Job Type	Bulk V1	Status	Closed
Submitted By	Jaya Kushwaha	Operation	Insert	Total Processing Time (ms)	428
Start Time	9/25/2025, 6:55 PM IST	Queued Batches	0	API Active Processing Time (ms)	380
End Time	9/25/2025, 6:55 PM IST	In Progress Batches	0	Apex Processing Time (ms)	308
Time to Complete ([hh]:mm:ss)	00:02	Completed Batches	1		
Object	Appointment	Failed Batches	0		
External ID Field		Progress	100%		
Content Type	CSV	Records Processed	25		
Concurrency Mode	Parallel	Records Failed	0		
API Version	64.0	Retries	0		
<div>Reload</div>					

Batches

View Request	View Result	Batch ID	Start Time	End Time	Total Processing Time (ms)	API Active Processing Time (ms)	Apex Processing Time (ms)	Records Processed	Records Failed	Retry Count	State Message	Status
View Request	View Result	751gL00000BFGSc	9/25/2025, 6:55 PM	9/25/2025, 6:55 PM	428	380	308	25	0	0		Completed

3. VS Code & SFDX

Dev-friendly deployments and user-friendly platforms such as VS code is used to integrate the apex programming and other work into the application so that it is seamless.

Phase 9: Reporting, Dashboards & Security Review

Goal: Monitor business & secure data.

1. Reports

Given is a report that shows the products in stock.

Home > Lightning Report

Report: Products Products Report			
	Product Name ▾	Product Description ▾	Product Code ▾
2	SLA: Silver	-	SL9040
3	GenWatt Propane 500kW	-	GC3040
4	SLA: Platinum	-	SL9080
5	GenWatt Propane 100kW	-	GC3020
6	GenWatt Propane 1500kW	-	GC3060
7	SLA: Bronze	-	SL9020
8	GenWatt Gasoline 750kW	-	GC5040
9	Installation: Portable	-	IN7020
10	SLA: Gold	-	SL9060
11	GenWatt Gasoline 300kW	-	GC5020
12	Installation: Industrial - Low	-	IN7040
13	GenWatt Gasoline 2000kW	-	GC5060
14	Installation: Industrial - Medium	-	IN7060
15	Fabrics	-	F01
16	Hair Gel	Hair gel	HG01

2. Dashboards

Dashboard for product availability is created for easy access of the details provided in the report.

Product Dashboard

Products Report

Product Name ↑	Product Description	Product Code
Fabrics	-	F01
GenWatt Gasoline 2000kW	-	GC5060
GenWatt Gasoline 300kW	-	GC5020
GenWatt Gasoline 750kW	-	GC5040
GenWatt Propane 100kW	-	GC3020
GenWatt Propane 1500kW	-	GC3060
GenWatt Propane 500kW	-	GC3040

View Report (Products Report)