Phase 6: User Interface Development

PetCare CRM

A Salesforce-Based Pet Shop Management System for Customer Engagement and Service Automation

Goal: Make the Pet Care CRM user-friendly for admins, staff, and pet owners, ensuring easy access to pets, bookings, and services.

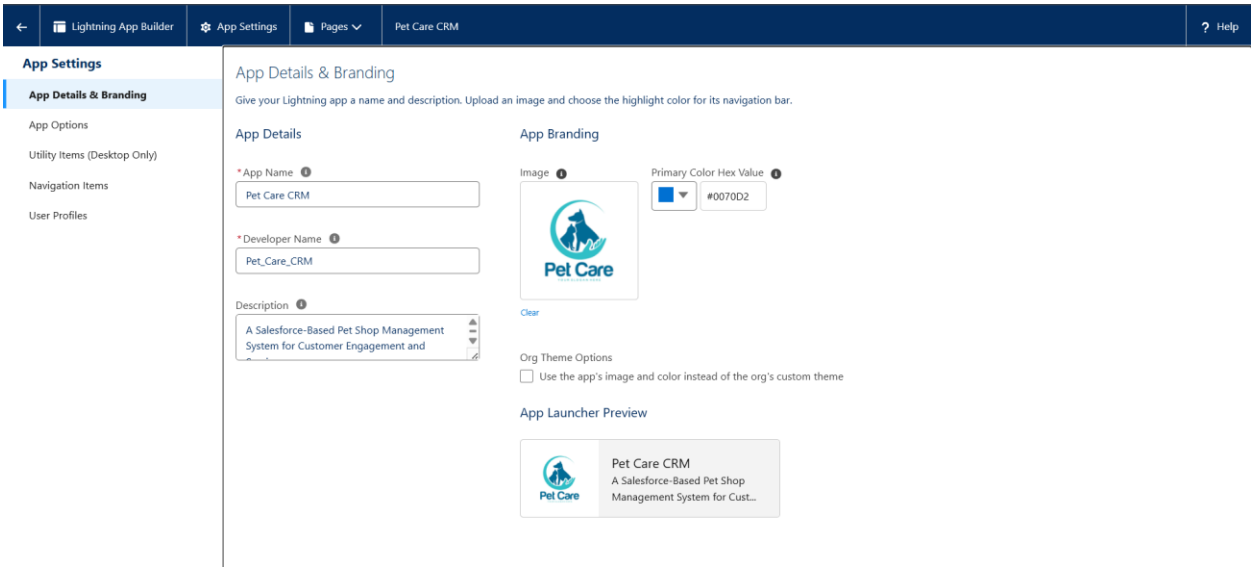
1) **Lightning App Builder – Create Pet Care CRM App**

Explanation:
The Lightning App Builder allows you to create a custom Salesforce app tailored to your Pet Care CRM. It groups all the relevant objects, pages, and features into a single interface, making it easier for users to navigate.

Steps:

1. Go to **Setup → App Manager → New Lightning App**.
2. Enter **App Name:** Pet Care CRM.
3. Add a **logo, color theme, and description** to match branding.
4. Assign **user profiles** such as Admin and Staff to control access.
5. Add the necessary **Tabs** for Pets and Bookings (see next step).
6. Finish the wizard → the app is now ready to launch.

Purpose: Provides a centralized app for managing pets, bookings, and services.



2)Record Pages – Pet/Service Booking Pages

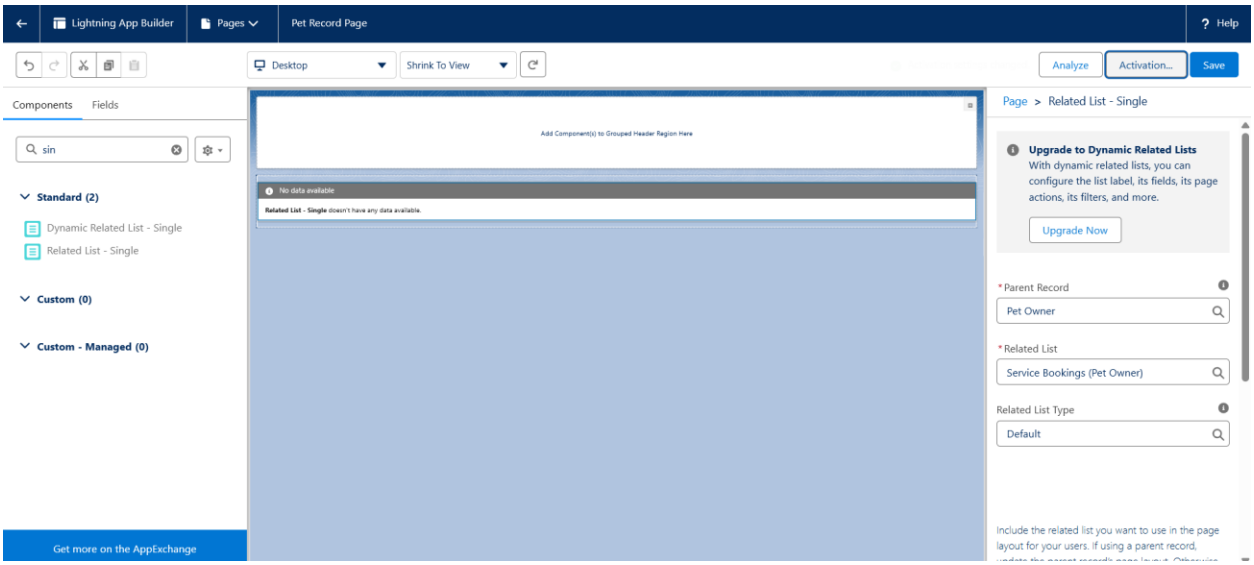
Explanation:

Record pages display detailed information for each pet or service booking. Customizing them ensures users can easily view all related bookings without navigating through multiple pages.

Steps:

1. Go to **Setup → Object Manager → Pet__c → Lightning Record Pages → New**.
2. Choose **Record Page type → Start from App Default**.
3. Drag **Related Lists – Single** component onto the page.
4. Configure it to show **Bookings related to that pet**.
5. Save and **activate the page** for your app.

Purpose: Makes it easy to see all bookings related to a pet in one place.



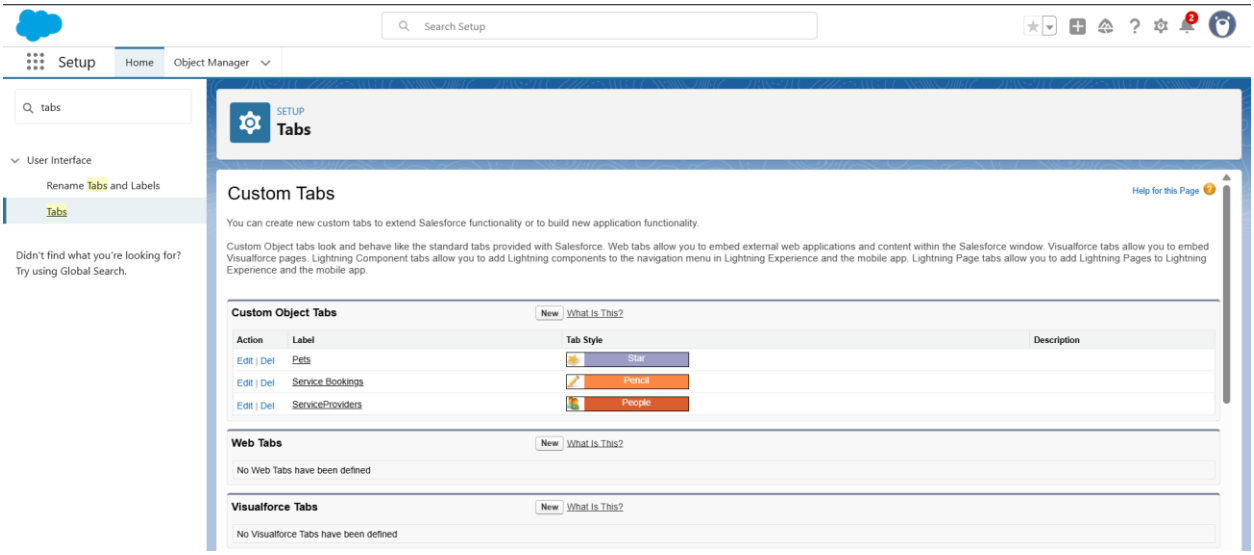
3)Tabs – Pets & Bookings

Explanation:
Tabs provide easy navigation within the app, allowing users to quickly access objects like Pets and Service Bookings.

Steps:

1. Go to **Setup → Tabs → New**.
2. Create a **Custom Object Tab** for:
    - Pet__c → Label: **Pets**
    - Service_Booking__c → Label: **Bookings**
3. Add these tabs to your **Pet Care CRM App** via the App Manager.

Purpose: Ensures users can easily switch between Pets and Bookings without leaving the app.



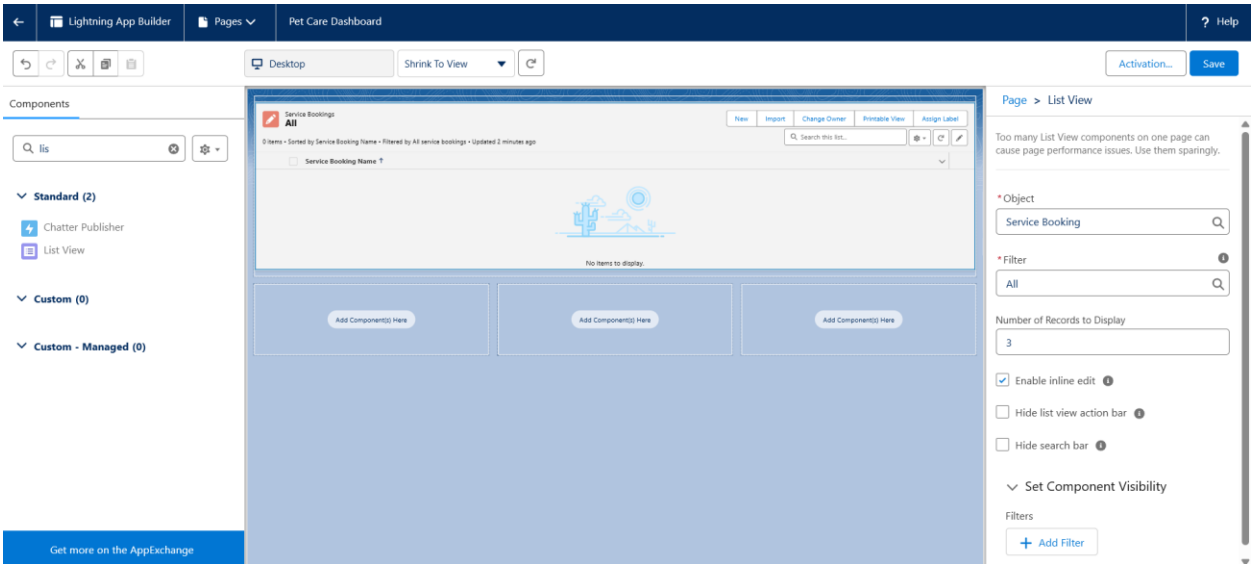
4) Home Page Layout – Dashboard

Explanation:
The Home Page acts as a dashboard, giving users an overview of key metrics such as today's bookings, upcoming appointments, and overdue bookings.

Steps:

1. Go to **Setup → Lightning App Builder → Home Page → New**.
2. Drag **Report Chart** or **List View** components onto the page:
    - Show **Today's Bookings**
    - Show **Upcoming Bookings**
    - Show **Overdue Bookings**
3. Save and **activate the page** as the default home page for your app users.

Purpose: Helps staff and managers quickly monitor bookings and service utilization.



5)Utility Bar – Quick New Booking (Optional)

Explanation:
The Utility Bar allows users to quickly access common actions, like creating a new booking, from any page in the app.

Steps:

1. Go to **Setup → App Manager → Edit Pet Care CRM App → Utility Bar → Add**.
2. Select **Custom Action → New Booking**.
3. Give a label: New Booking.
4. Save and finish.

6) LWC – Search Pets / Services

Explanation:
Lightning Web Components (LWC) provide dynamic and interactive UI components. A search component allows users to find available services for pets based on selected dates.

Steps:

1. Create an LWC named searchAvailableServices.
2. Include the following elements:
    - **Date input fields** for start and end date

- **Search button**
- **Data table** to display search results

Example HTML snippet:

```
<lightning-input type="date" label="Start Date" onchange={handleStartDate}></lightning-input>

<lightning-input type="date" label="End Date" onchange={handleEndDate}></lightning-input>

<lightning-button label="Search" onclick={searchServices}></lightning-button>


<lightning-datatable
   data={services}
   columns={columns}
   key-field="Id">
</lightning-datatable>
```

Purpose: Improves user experience by providing an interactive search interface for available services.

---

7)Apex with LWC – Imperative Call

Explanation:
Imperative Apex calls allow the LWC to fetch data dynamically when a user interacts with the UI, such as clicking the Search button.

Steps:

1. Create an Apex method in BookingService with @AuraEnabled(cacheable=true):

```
@AuraEnabled(cacheable=true)
public static List<Service__c> getAvailableServices(Date startDate, Date endDate){
   return [SELECT Id, Name FROM Service__c WHERE Status__c = 'Available'];
}
```

2. Call this method imperatively from the LWC when the user clicks Search.

Purpose: Fetches real-time data from Salesforce to display available services.

8)Events in LWC – Child to Parent

Explanation:
Custom events allow child LWCs (e.g., search form) to communicate with parent LWCs (e.g., datatable for results).

Steps:

1. In the child LWC, create a **CustomEvent**:

```
handleSearch() {
   const searchEvent = new CustomEvent('searchresults', { detail: this.selectedDates });
   this.dispatchEvent(searchEvent);
}
```

2. In the parent LWC, listen for the event:

```
<c-search-form onsearchresults={handleResults}></c-search-form>
```

3. Update the datatable using the handleResults method.

Purpose: Ensures modular components can communicate and update the UI dynamically.

9) Wire Adapters – Display Available Services

Explanation:
Wire adapters allow components to automatically fetch and display Salesforce data reactively.

Steps:

```
@wire(getAvailableServices, { startDate: '$startDate', endDate: '$endDate' })
services;
```

- The data table updates automatically whenever the user changes start or end date.

Purpose: Provides a real-time display of available services without manual refresh.

10) Imperative Apex Calls – Book Now

Explanation:
When a user clicks the **Book Now** button, an imperative Apex call creates a new booking record in Salesforce.

Steps:

```
bookService(serviceId) {
   createBooking({ serviceId: serviceId })
      .then(result => { /* Navigate to booking page */ })
      .catch(error => { console.error(error); });
}
```

- The createBooking method is an Apex function with @AuraEnabled.

Purpose: Allows users to create bookings directly from the UI.

11)Navigation Service – Go to Booking Record

Explanation:
After creating a booking, navigating the user to the newly created **Booking record page** provides a smooth workflow.

Steps:

```
import { NavigationMixin } from 'lightning/navigation';


this[NavigationMixin.Navigate]({
   type: 'standard__recordPage',
   attributes: {
      recordId: bookingId,
      objectApiName: 'Service_Booking__c',
      actionName: 'view'
   }
});
```

Purpose: Improves usability by automatically directing users to the relevant booking record