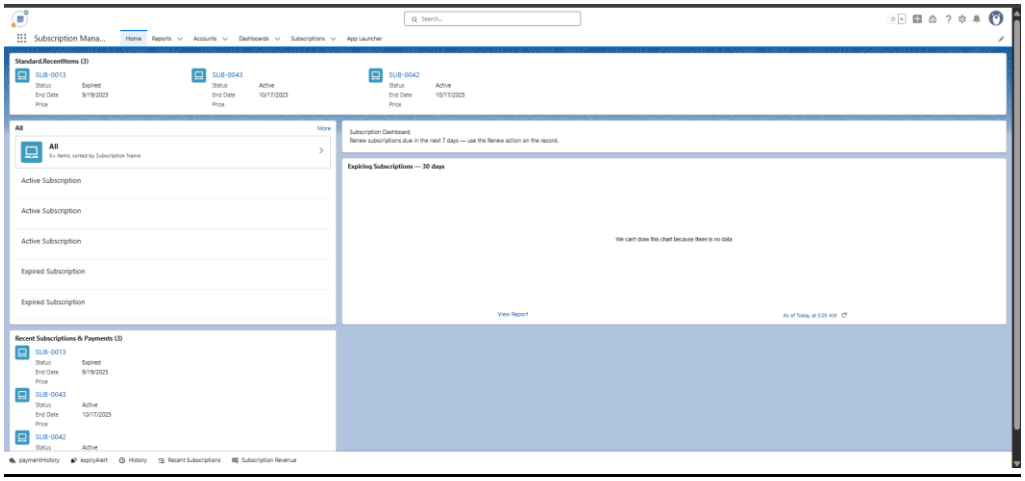
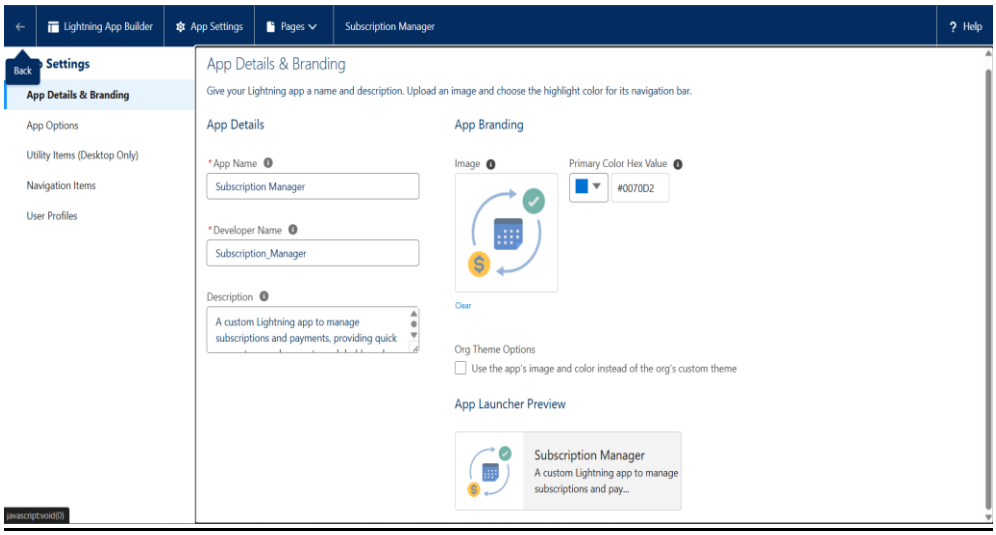
Project Title – “Smart Subscription Tracker”

Phase 6: User Interface Development

Lightning App Builder

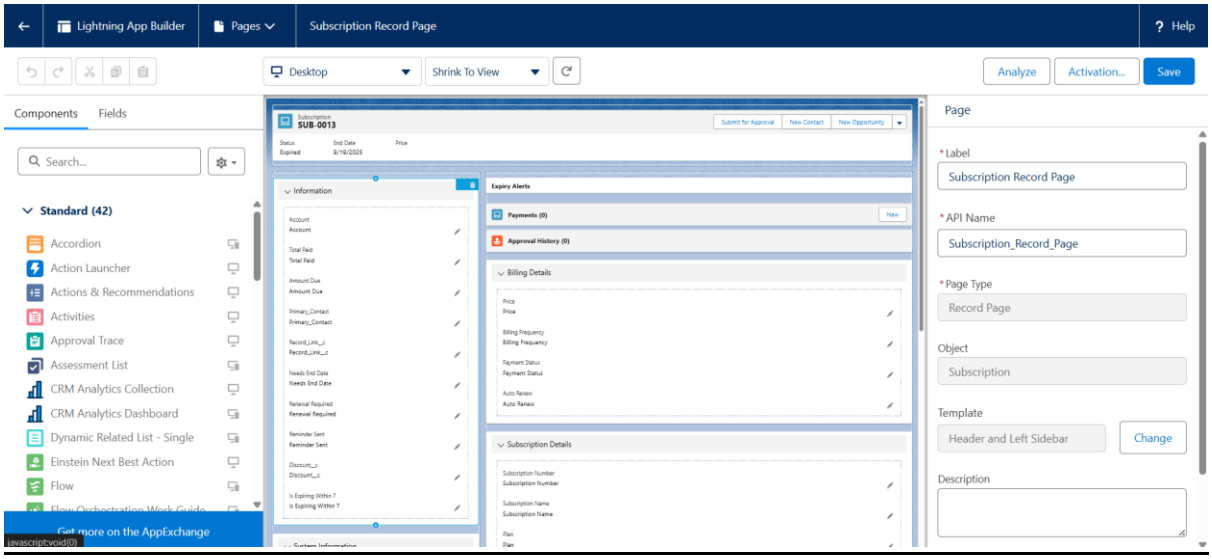
- Setup → **App Manager** → **New Lightning App**.
- Fill: **App Name** Subscription Manager, developer name auto-fills, short description.
- Branding: upload logo and blue app color.
- Choose **Standard Navigation** (unless you want console behavior.
- Finish wizard and **Save**.





Record Pages

- Setup → Lightning App Builder → New → Record Page → name Subscription Record - Custom.
- Select template (Header + Two Columns recommended). Choose object Subscription__c (or Subscription_c__c per your org).
- Drag these components into slots:
- Top: Path (if you have subscription statuses), Highlights Panel.
- Left column: Record Detail (or Dynamic Forms field sections so you can move fields around).
- Right column: Custom LWCs:
    - ExpiryAlert (shows days left, calls attention when < X days)
    - RenewButton (imperative Apex call to renew; visibility only when Status = Active and End_Date__c <= TODAY + 7)
    - PaymentHistory (related list or LWC with payment chart)
- Below: Related Lists component (show Payments, Activities).
- Save → Activate. Choose activation scope: App Default for Subscription Manager OR Org Default for the record type and profiles you prefer.



Tabs

1. From **Available Items**, select what you want to appear in your app:

Recommended Tabs for Subscription Tracker:

- **Subscriptions** → Your Subscription__c object.
- **Accounts** → If subscriptions are linked to accounts.
- **Payments** → Any related Payment object.
- **Reports** → For subscription analytics (charts, dashboards).
- **Custom Tab** → If you have a custom LWC (like Subscription Expiry Alerts) deployed as a tab.

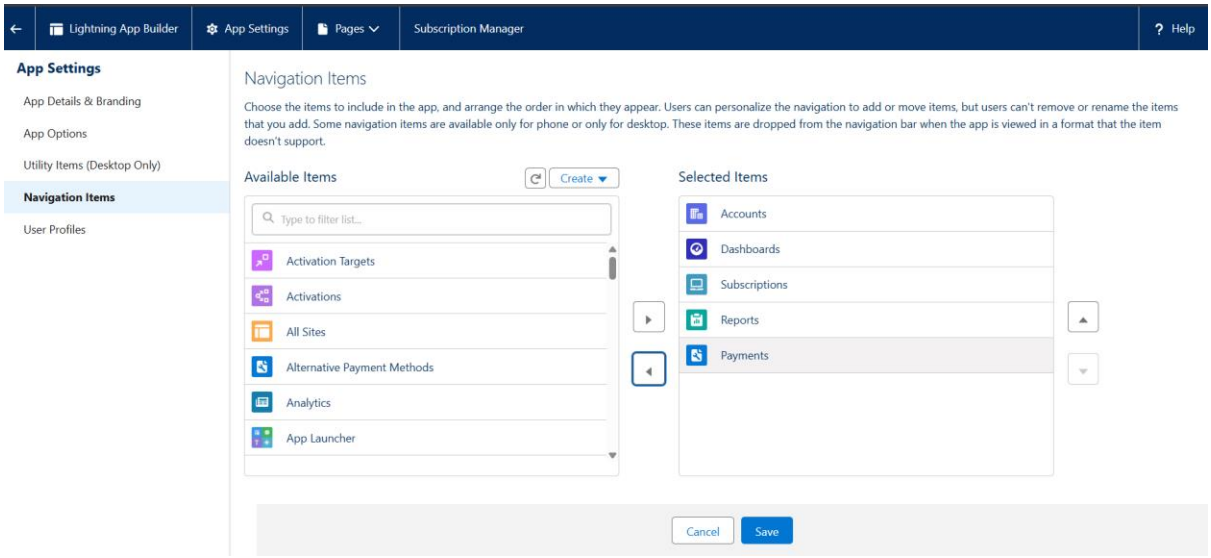
2. Click **Add** → Moves it to **Selected Items**.

Step 5: Arrange Tabs

- Use **Up/Down arrows** to arrange the tabs in the order you want:
  - Example:
    1. Subscriptions
    2. Accounts
    3. Payments
    4. Reports
    5. Alerts (Custom LWC Tab)

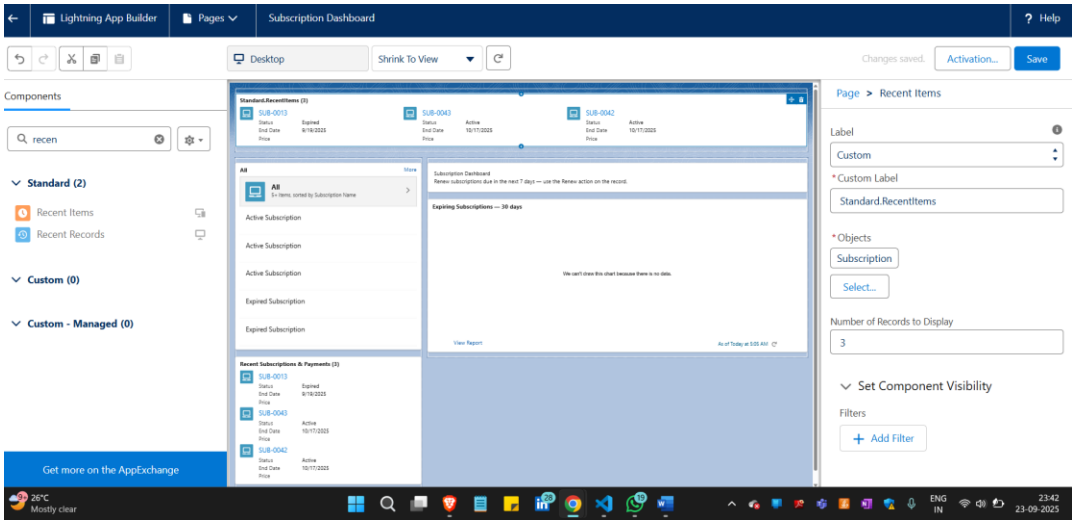
Step 6: Save App

1. After arranging tabs, click **Next** → then **Save & Finish**.



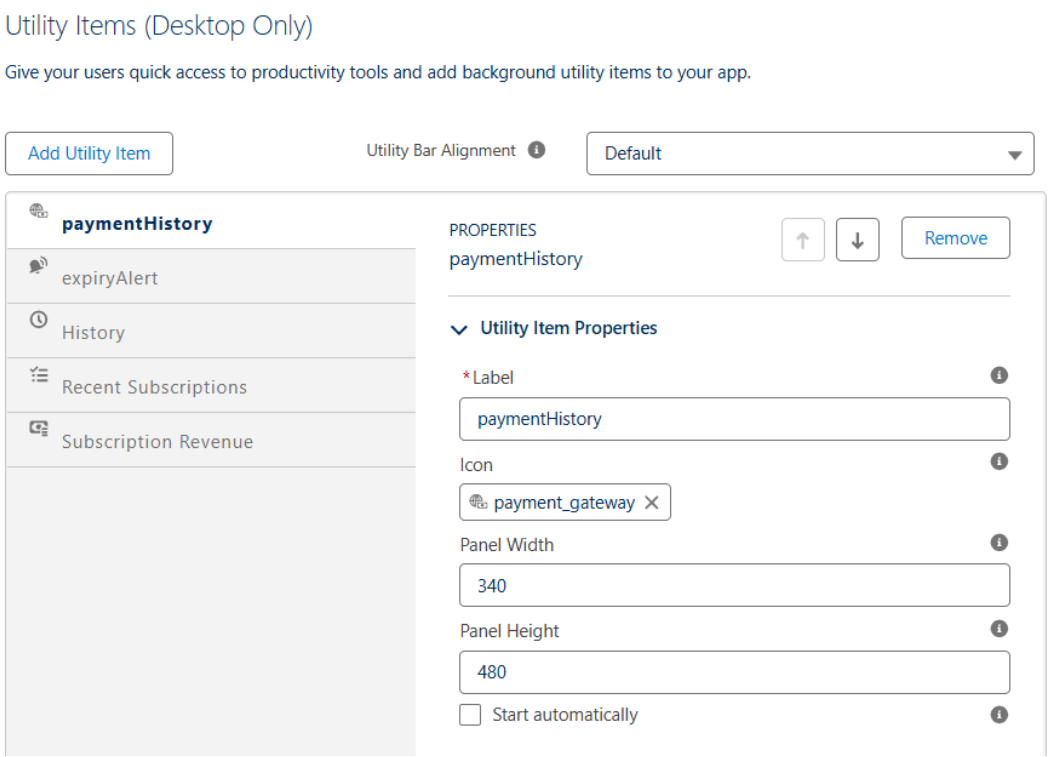
Home Page Layouts

- Go to Setup → Lightning App Builder → New.
- Select Home Page (dashboard) or App Page (custom app tab).
- Pick a layout template (e.g., Two-Column).
- Name the page (e.g., *Subscription Home* / *Subscription Dashboard*).
- Add Components:
- Report Chart – subscription stats.
- List View – list of subscriptions/accounts.
- Recent Items – quick access.
- Custom LWC – e.g., subscription expiry alert.
- Arrange/resize components as needed.
- Save & Activate → Assign to the Subscription Manager app



🧩 Utility Bar

- Utility Bar Setup
- Enable Utility Bar → In App Editor, click Utility Bar → Add Utility Item.
- Add Items (examples for Subscription Project):
  - Recent Subscriptions → quick access to active subscriptions.
  - Payment/Expiry Alerts → reminders for upcoming expirations.
  - Report Charts → mini-dashboard of revenue or subscription status.
  - Custom LWC (optional) → KPI panel or notifications.
- Configuration → Select type (Standard Component / LWC), set Label, Icon, and save.



LWC (Lightning Web Components)

➢ LWC 1: ExpiryAlert

Html-code

force-app > main > default > lwc > expiryAlert > expiryAlert.html > ...

```
<template>
    <lightning-card title="Expiry Alerts">
        <template if:true={subscriptions}>
            <template for:each={subscriptions} for:item="sub">
                <p key={sub.Id}>
                    {sub.Name} expires in {sub.daysLeft} days
                </p>
            </template>
        </template>
        <template if:true={error}>
            <p class="slds-text-color_error">{error}</p>
        </template>
    </lightning-card>
</template>
```

Xml-code

force-app > main > default > lwc > expiryAlert > expiryAlert.js-meta.xml > LightningComponentBundle > targets

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>59.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>

        <target>lightning__HomePage</target>
        <target>lightning__RecordPage</target>
        <target>lightning__AppPage</target>
        <target>lightning__UtilityBar</target>
    </targets>
</LightningComponentBundle>
```

Js-code

```
import { LightningElement, api, wire } from 'lwc';
import getDaysUntilEnd from '@salesforce/apex/SubscriptionController.getDaysUntilEnd';
import getPaymentsForSubscription from '@salesforce/apex/SubscriptionController.getPaymentsForSubscri

Windsurf: Refactor | Explain
export default class ExpiryAlert extends LightningElement {
    subscriptions;
    error;

    @api recordId; // optional if used on record page

    Windsurf: Refactor | Explain | Generate JSDoc | X
    connectedCallback() {
        // Example: fetch subscriptions with expiry in next 7 days
        // You can customize this later to call an Apex method returning multiple subscriptions
    }
}
```

➢ LWC 2: Payment History

Html-code

```
<template>
    <lightning-card title="Payment History">
        <template if:true={payments}>
            <template for:each={payments} for:item="pay">
                <p key={pay.Id}>
                    {pay.Name} | {pay.Payment_Date__c} | Amount: {pay.Amount__c}
                </p>
            </template>
        </template>
        <template if:true={error}>
            <p class="slds-text-color_error">{error}</p>
        </template>
    </lightning-card>
</template>
```

Xml-code

force-app > main > default > lwc > paymentHistory > paymentHistory.js-meta.xml > ...

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>59.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__HomePage</target>
        <target>lightning__RecordPage</target>
        <target>lightning__AppPage</target>
        <target>lightning__UtilityBar</target>
    </targets>
</LightningComponentBundle>
```

Js-code

```
import { LightningElement, api, wire } from 'lwc';
import getPaymentsForSubscription from '@salesforce/apex/SubscriptionController.getPaymentsForSubscri

Windsurf: Refactor | Explain
export default class PaymentHistory extends LightningElement {
    @api recordId;
    payments;
    error;

    Windsurf: Refactor | Explain | Generate JSDoc | ×
    @wire(getPaymentsForSubscription, { subscriptionId: '$recordId' })
    wiredPayments({ error, data }) {
        if (data) {
            this.payments = data;
            this.error = undefined;
        } else if (error) {
            this.error = error.body ? error.body.message : error.message;
            this.payments = undefined;
        }
    }
}
```

➢ LWC 3: Renew Subscription

Html-code

```
1   <template>
2       <lightning-card title="Renew Subscription">
3           <lightning-button label="Renew Now" onclick={handleRenew} class="slds-m-around_small"></lightning-button>
4           <t
5
6           </
7           <t
8
9           </template>
10      </lightning-card>
11  </template>
12
```

The template element is used to declare fragments of HTML that can be cloned and inserted in the document by script.

Widely available across major browsers (Baseline since 2015)

MDN Reference

Xml-code

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>59.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__HomePage</target>
        <target>lightning__RecordPage</target>
        <target>lightning__AppPage</target>
        <target>lightning__UtilityBar</target>
    </targets>
</LightningComponentBundle>
```

Js-code

force-app > main > default > lwc > renewButton > renewButton.js > ...

```js
import { LightningElement, api } from 'lwc';
import renewSubscription from '@salesforce/apex/SubscriptionController.renewSubscription';

Windsurf: Refactor | Explain
export default class RenewButton extends LightningElement {
    @api recordId; // Subscription record Id from Record Page
    message;
    error;

    Windsurf: Refactor | Explain | Generate JSDoc | ×
    handleRenew() {
        this.message = '';
        this.error = '';

        renewSubscription({ subscriptionId: this.recordId, extensionDays: 30 })
            .then(result => {
                this.message = 'Subscription renewed! New end date: ' + result;
            })
            .catch(error => {
                this.error = error.body ? error.body.message : error.message;
            });
    }
}
```

Apex controller

```apex
public with sharing class SubscriptionController {

    // LWC 1: ExpiryAlert
    @AuraEnabled(cacheable=true)
    public static Integer getDaysUntilEnd(Id subscriptionId) {
        if (subscriptionId == null) return null;

        Subscription__c sub = [
            SELECT End_Date__c
            FROM Subscription__c
            WHERE Id = :subscriptionId
            LIMIT 1
        ];

        if (sub.End_Date__c == null) return null;

        return Date.today().daysBetween(sub.End_Date__c);
    }

    // LWC 3: PaymentHistory
    @AuraEnabled(cacheable=true)
    public static List<Payment__c> getPaymentsForSubscription(Id subscriptionId) {
        if (subscriptionId == null) return new List<Payment__c>();

        return [
            SELECT Id, Name, Amount__c, Payment_Date__c, Status__c
            FROM Payment__c
            WHERE Subscription__c = :subscriptionId
            ORDER BY Payment_Date__c DESC
            LIMIT 50
        ];
    }

    // LWC 2: RenewButton
    @AuraEnabled
    public static String renewSubscription(Id subscriptionId, Integer extensionDays) {
        if (subscriptionId == null) {
            throw new AuraHandledException("Subscription Id is required.");
        }
        if (extensionDays == null || extensionDays <= 0) {
            extensionDays = 30; // default
        }

        try {
            // Get subscription
            Subscription__c s = [
                SELECT Id, End_Date__c
                FROM Subscription__c
                WHERE Id = :subscriptionId
                LIMIT 1
                FOR UPDATE
            ];

            if (s.End_Date__c == null) {
                s.End_Date__c = Date.today();
            }

            // Extend subscription
            Date newEnd = s.End_Date__c.addDays(extensionDays);
            s.End_Date__c = newEnd;
            update s;

            // Create a payment record
            Payment__c p = new Payment__c(
                Subscription__c = s.Id,
                Payment_Date__c = Date.today(),
                Amount__c = 100,    // adjust as per your project
                Status__c = 'Paid'  // make sure 'Paid' exists in picklist
            );
            insert p;

            return newEnd.format();
        } catch (Exception e) {
            throw new AuraHandledException("Unable to renew subscription: " + e.getMessage());
        }
    }
}
```

🧩 Apex Controller (SubscriptionController)

- **Purpose**: Provides server-side logic for Lightning Web Components (LWCs) used in the project.
- **Key Features Implemented**:
    1. **ExpiryAlert** → Method getDaysUntilEnd() calculates remaining days until subscription expiry.
    2. **PaymentHistory** → Method getPaymentsForSubscription() fetches the list of recent payments linked to a subscription.
    3. **RenewButton** → Method renewSubscription() extends subscription end date and creates a new payment record automatically.
- **Error Handling**: Uses AuraHandledException to provide user-friendly error messages to the frontend.
- **Reusability**: Designed as a single controller class so multiple LWCs (ExpiryAlert, PaymentHistory, RenewButton) can access it.
- **Business Logic**: Ensures subscription renewal not only extends the date but also logs a related payment record.
- **Data Security**: Marked as with sharing so that record-level sharing rules are respected.
- **Scalability**: Queries and updates are limited with best practices (LIMIT, FOR UPDATE) to avoid errors in concurrent use.
