

Phase 6: User Interface Development (IP & Patent Management)

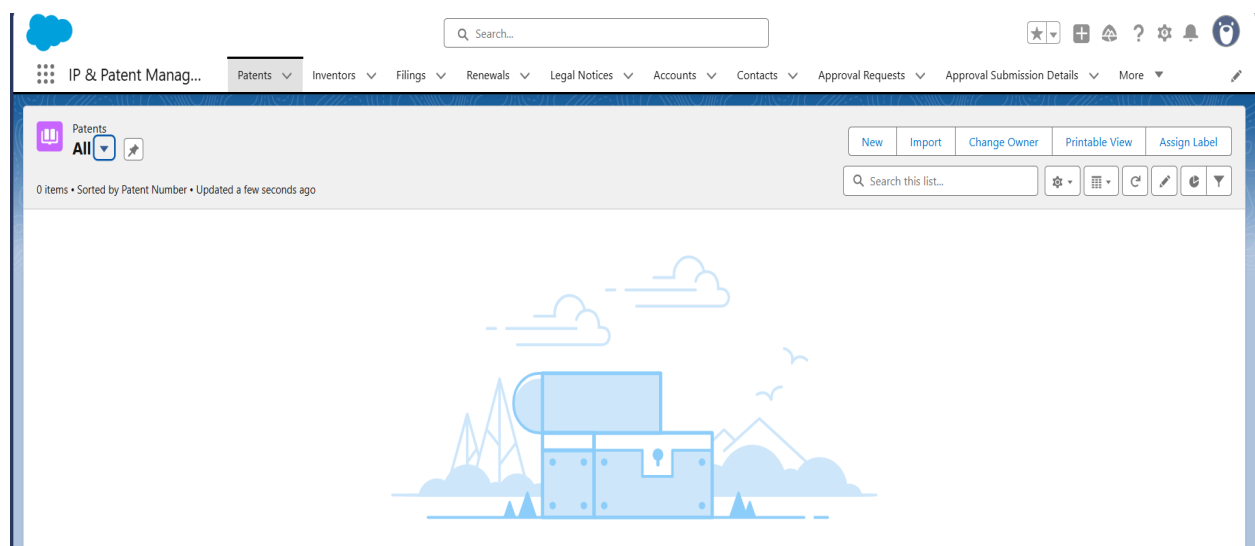
1. Lightning App Builder

What we did:

The **IP & Patent Management** Lightning App was created using Lightning App Builder. The app contains navigation for the main objects (Patents, Inventors, Filings, Renewals, Legal Notices) and the custom Home and Record pages. App branding and icon were set to “IP” style and the app is assigned to the IP Admin and Legal Officer profiles for testing.

Steps implemented (short):

1. Setup → App Manager → New Lightning App → Name: *IP & Patent Management App*.
2. Set logo, description, and choose navigation style (Standard).
3. Add navigation items: Patents, Inventors, Filings, Renewals, Reports, Dashboards.
4. Assign app to profiles: IP Admin, Legal Officer.
5. Save & Finish.



2. Record Pages

- The **Patent Record Page** was customized to display the patent summary and all related items in one view: Patent Number, Title, Status, Filing Date, Grant Date, Expiry Date, **Days Until Expiry** formula, Related Filings list, Related Renewals list, Related Legal Notices, and the **Patent Timeline** LWC (custom) placed in the right-hand region for quick chronological view. Quick Actions (Submit for Approval, Create Renewal, Send to Legal) were added to the page header.

New Patent: Utility Patent

* = Required Information

Information

* Patent Number

Assigned Approver (Legal Officer)

Title

Abstract

Priority Date

Expiry Date

Owner

Parvez Sharief

Search People...

Filing Date

Reminder 90 Sent

Reminder 30 Sent

Reminder 7 Sent

Potential Duplicate

Status

--None--

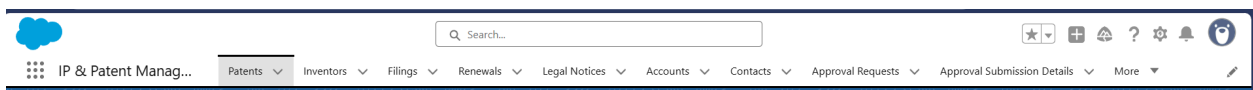
Cancel Save & New Save

3. Tabs

Custom tabs were created for the main objects so users can navigate quickly from the App navbar. Tabs: Patent, Inventor, Filing, Renewal, Legal Notice, and a Reports tab for dashboards.

Steps implemented:

1. Setup → Tabs → New → Custom Object Tabs → select Patent (choose icon) → Save.
2. Repeat for Inventor, Filing, Renewal, Legal Notice.
3. Add tabs to IP & Patent Management App navigation.

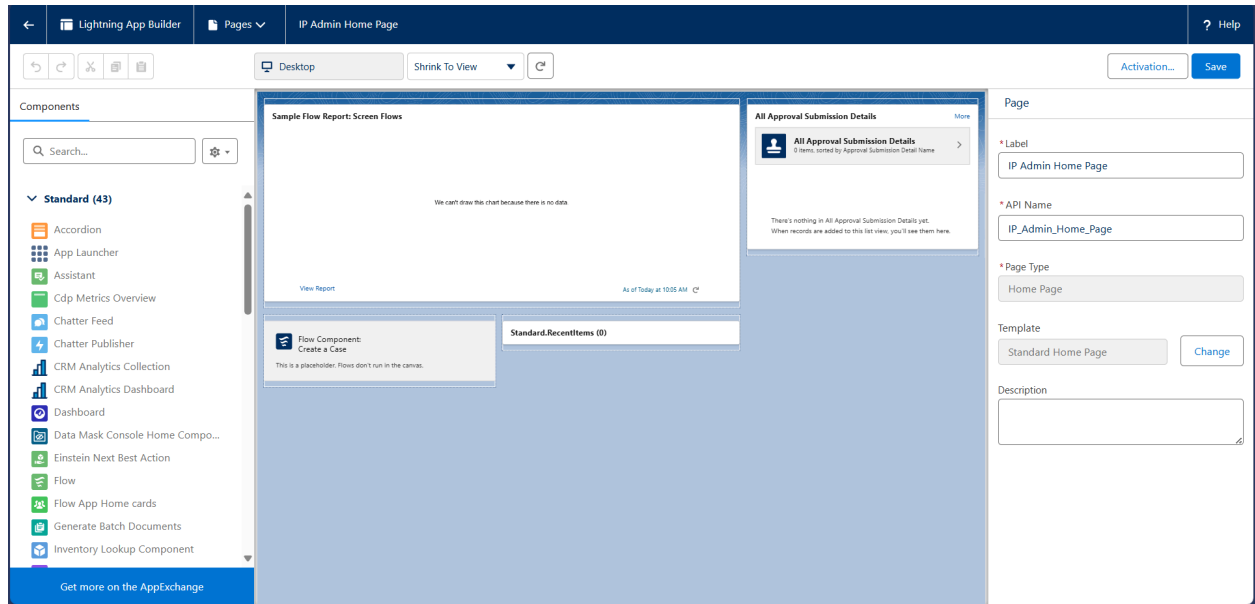


4. Home Page Layouts

- A role-specific **Home Page** for IP Admins was created showing: Upcoming Renewals (report chart), Pending Approvals (list view), Fast Create (flow component for quick patent submission), and Recent Activity. The Home was assigned to IP Admin profile so demos show immediate business value on login.

Steps implemented:

1. Setup → Lightning App Builder → Home Page → New.
2. Add components: Report Chart (Upcoming Renewals), List View (Pending Approval Patents), Flow (Quick Create Patent), Recent Items.
3. Save & Assign to IP Admin profile.



5. Utility Bar

A minimal Utility Bar was added to the App to give quick access to **New Patent (Quick Action)** and **Patent Search**. This speeds up data entry during demos. Full advanced utilities (console integrations) were not required.

Steps implemented:

1. App Manager → Edit IP App → Utility Bar → Add: *Global Quick Action (New Patent), Recent Items*.

6. Lightning Web Components (LWC)

- We implemented one focused LWC: **Patent Timeline**. This component fetches Filings and Renewals for the current Patent record and displays them in chronological order. This LWC gives a clear, modern UI element that demonstrates LWC + Apex integration and is small enough to finish quickly.

patentTimeline.html:

```

patentTimeline.js  patentTimeline.html  patentTimeline.js-meta.xml  PatentController.cls 5  PatentController.cls-meta.xml
patentTimeline.js
1  <!--
2  @description      :
3  @author           : ChangeMeIn@UserSettingsUnder.SFDoc
4  @group            :
5  @last modified on : 10-03-2025
6  @last modified by : ChangeMeIn@UserSettingsUnder.SFDoc
7  -->
8  <template>
9  <lightning-card title="Patent Timeline">
10   <template if:true={timeline}>
11     <ul class="slds-timeline">
12       <template for:each={timeline} for:item="item">
13         <li key={item.recordId} class="slds-timeline__item">
14           <div class="slds-timeline__item_detail">
15             <p class="slds-text-body_regular">
16               <a data-record-id={item.recordId} href="javascript:void(0);" onclick={handleOpenRecord}>
17                 {item.eventDate} - {item.type}: {item.title} [{item.status}]
18               </a>
19             </p>
20           </div>
21         </li>
22       </template>
23     </ul>
24   </template>
25   <template if:true={error}>
26     <div class="slds-text-color_error">Error loading timeline</div>
27   </template>
28 </lightning-card>
29 </template>
30

```

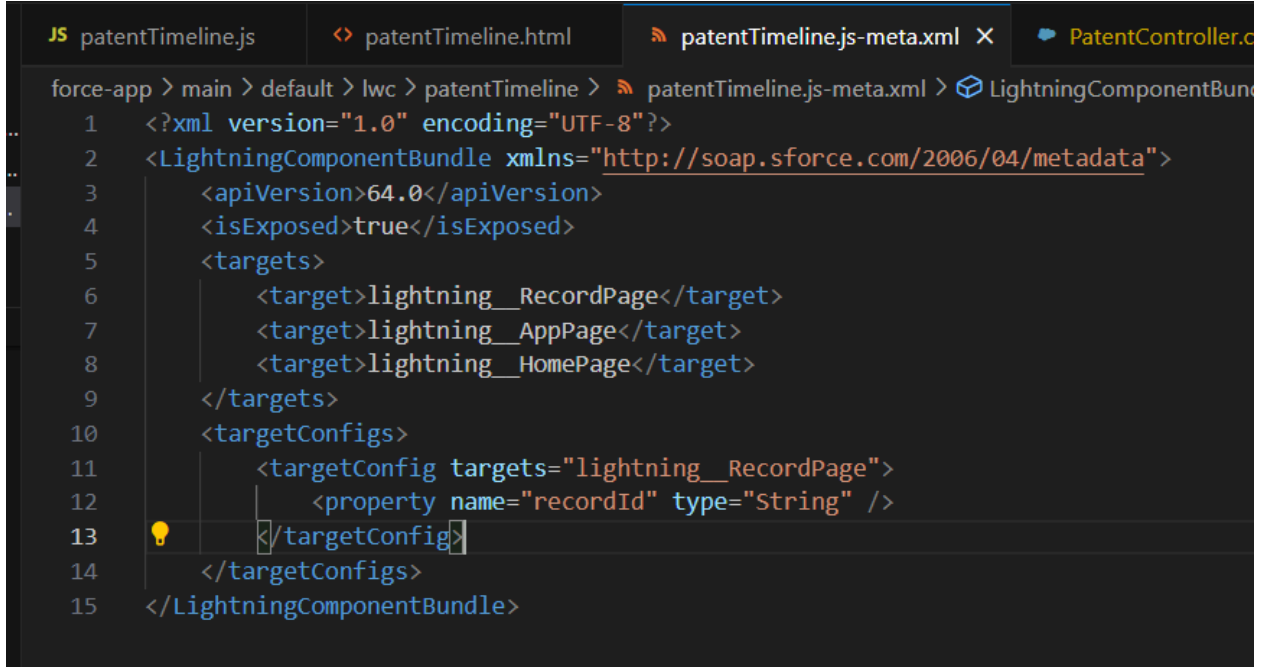
patentTimeline.js:

```

JS patentTimeline.js  patentTimeline.html  patentTimeline.js-meta.xml  PatentController.cls 5  PatentController.cls
force-app > main > default > lwc > patentTimeline > JS patentTimeline.js > ...
1  import { LightningElement, api, track, wire } from 'lwc';
2  import getTimeline from '@salesforce/apex/PatentController.getTimeline';
3
4  export default class PatentTimeline extends LightningElement {
5    @api recordId;
6    @track timeline = [];
7    @track error;
8
9    @wire(getTimeline, { patentId: '$recordId' })
10   wiredTimeline({ error, data }) {
11     if (data) {
12       // sort by date descending (most recent first)
13       this.timeline = data.slice().sort((a,b) => new Date(b.when) - new Date(a.when));
14       this.error = undefined;
15     } else if (error) {
16       this.error = error;
17       this.timeline = [];
18     }
19   }
20
21   handleOpenRecord(event) {
22     const recId = event.currentTarget.dataset.recordId;
23     this.dispatchEvent(new CustomEvent('openrecord', { detail: recId }));
24   }
25 }
26

```

[patentTimeline.js](#)-meta.xml:



```
JS patentTimeline.js  <> patentTimeline.html  patentTimeline.js-meta.xml X  PatentController.c
force-app > main > default > lwc > patentTimeline > patentTimeline.js-meta.xml > LightningComponentBund
1  <?xml version="1.0" encoding="UTF-8"?>
2  <LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
3    <apiVersion>64.0</apiVersion>
4    <isExposed>true</isExposed>
5    <targets>
6      <target>lightning__RecordPage</target>
7      <target>lightning__AppPage</target>
8      <target>lightning__HomePage</target>
9    </targets>
10   <targetConfigs>
11     <targetConfig targets="lightning__RecordPage">
12       <property name="recordId" type="String" />
13     </targetConfig>
14   </targetConfigs>
15 </LightningComponentBundle>
```

7. Apex with LWC

- A lightweight Apex controller `PatentController` was created to return a simple timeline payload (combination of Filing and Renewal items). The Apex is cacheable for read operations and safe for LWC `@wire`. No heavy server processing is done — the Apex returns compact wrapper objects for easy consumption by the LWC.

```
JS patentTimeline.js  patentTimeline.html  patentTimelinejs-meta.xml  PatentController.cls 5  PatentController.cls-meta.xml
force-app > main > default > classes > PatentController.cls > ...
1 public with sharing class PatentController {
2     public class TimelineItem {
3         @AuraEnabled public String eventDate;
4         @AuraEnabled public String type;
5         @AuraEnabled public String title;
6         @AuraEnabled public String status;
7         @AuraEnabled public Id recordId;
8
9         public TimelineItem(String eventDate, String type, String title, String status, Id rid) {
10             this.eventDate = eventDate;
11             this.type = type;
12             this.title = title;
13             this.status = status;
14             this.recordId = rid;
15         }
16     }
17
18     @AuraEnabled(cacheable=true)
19     public static List<TimelineItem> getTimeline(Id patentId) {
20         List<TimelineItem> items = new List<TimelineItem>();
21         if (patentId == null) return items;
22
23         // Filings
24         for (Filing__c f : [SELECT Id, Filing_Date__c, Office__c, Status__c FROM Filing__c WHERE Patent__c = :patentId]) {
25             items.add(new TimelineItem(
26                 String.valueOf(f.Filing_Date__c),
27                 'Filing',
28                 f.Office__c == null ? 'Filing' : f.Office__c,
29                 f.Status__c,
30                 f.Id
31             ));
32         }
33
34         // Renewals
35         for (Renewal__c r : [SELECT Id, Renewal_Date__c, Renewal_Status__c FROM Renewal__c WHERE Patent__c = :patentId]) {
36             items.add(new TimelineItem(
37                 String.valueOf(r.Renewal_Date__c),
38                 'Renewal',
39                 'Renewal',
40                 r.Renewal_Status__c,
41                 r.Id
42             ));
43         }
44
45         return items;
46     }
47 }
48
```

8. Events in LWC

The `patentTimeline` LWC dispatches a **custom event** `openrecord` when a timeline item is clicked. The enclosing record page listens for this event (or the parent LWC can) and uses Navigation Service to open the selected record, enabling a simple and responsive UI navigation flow.

9. Wire Adapters

`@wire(getTimeline, { patentId: '$recordId' })` was used in the timeline LWC to fetch data reactively whenever the recordId changes (e.g., when user navigates to another patent). This ensures the timeline stays in sync with the current record.

Usage summary:

- Use `@wire` for read-only, reactive data calls (cacheable Apex or UI API adapters).
 - I used Apex `@AuraEnabled(cacheable=true)` methods wired into LWC for the timeline.
-

10. Imperative Apex Calls

I added an optional **imperative** Apex method `checkSimilarity` (mocked) on `PatentController` so that when user clicks “Check Similarity” in the LWC we can call that method and show results in a modal. This demonstrates how to call Apex on demand from LWC.

11. Navigation Service

Navigation Service (`NavigationMixin`) is used in the project to open related records from the timeline component. When a user clicks a timeline item, the component fires the `openrecord` event and the parent handles it with a `NavigationMixin.Navigate` call to open the record in view mode.