Phase 6: User Interface Development

Flight Reservation & Scheduling System

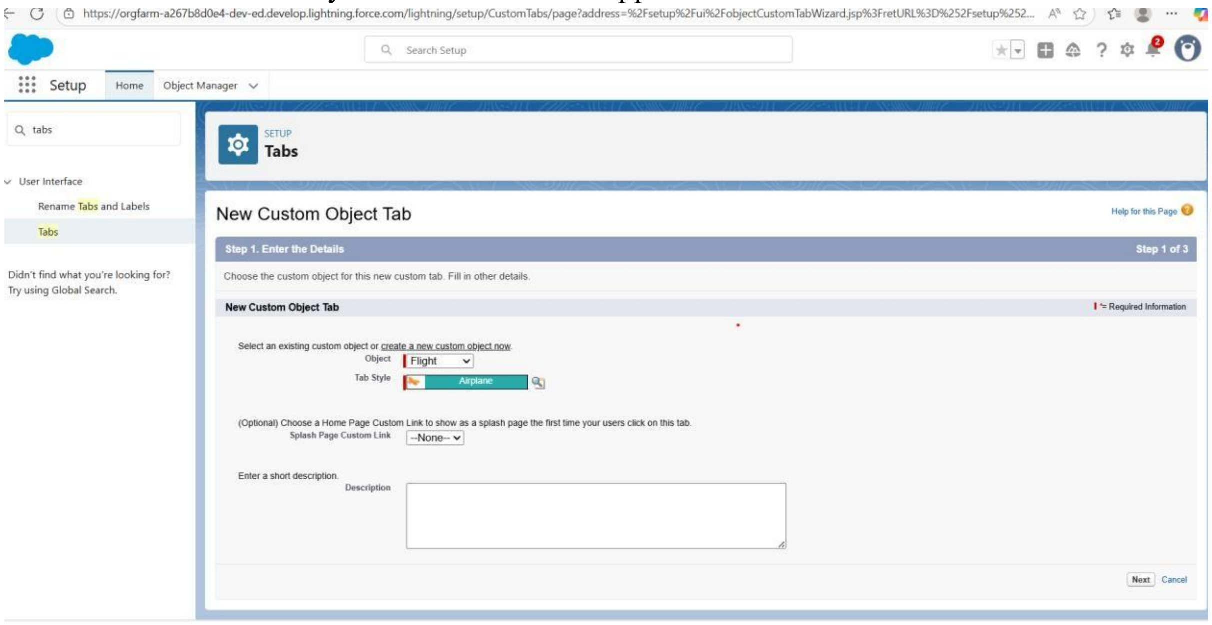
Salesforce-Based Flight Operations and Scheduling System

Step 1: Create a Lightning App

- Setup → **App Manager** → **New Lightning App**
- Name it Airline Console → choose navigation style → Save.

Step 2: Create Object Tabs

- Setup → **Tabs** → **New** → **Custom Object Tab** • Create tabs for Flight__c, Booking__c, Passenger__c.
- Add these tabs to your Airline Console app.



SETUP
Tabs

New Custom Object Tab

Step 1. Enter the Details — Step 1 of 3

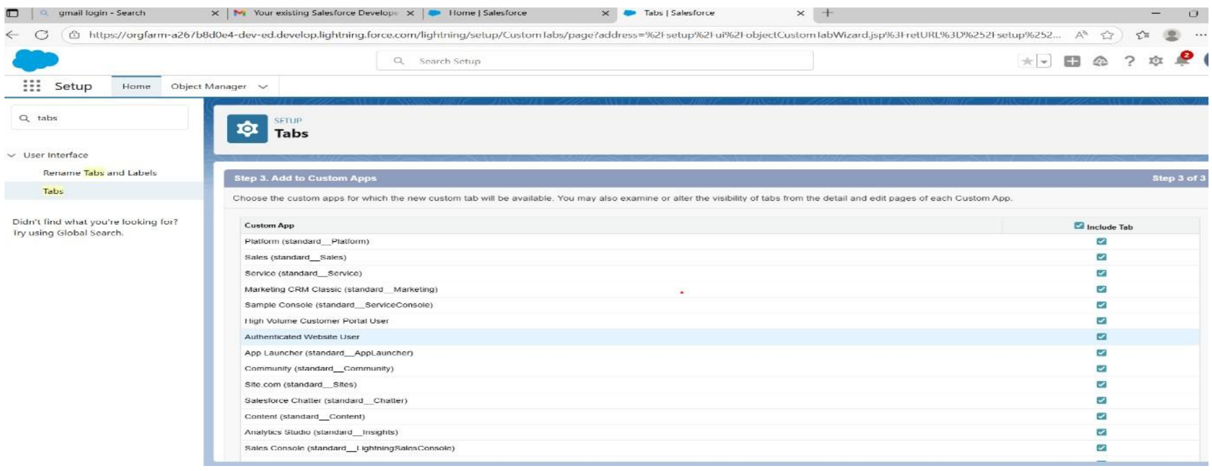
Choose the custom object for this new custom tab. Fill in other details.

New Custom Object Tab — | = Required Information

Select an existing custom object or create a new custom object now.
Object: Flight
Tab Style: Airplane

(Optional) Choose a Home Page Custom Link to show as a splash page the first time your users click on this tab.
Splash Page Custom Link: --None--

Enter a short description.
Description:

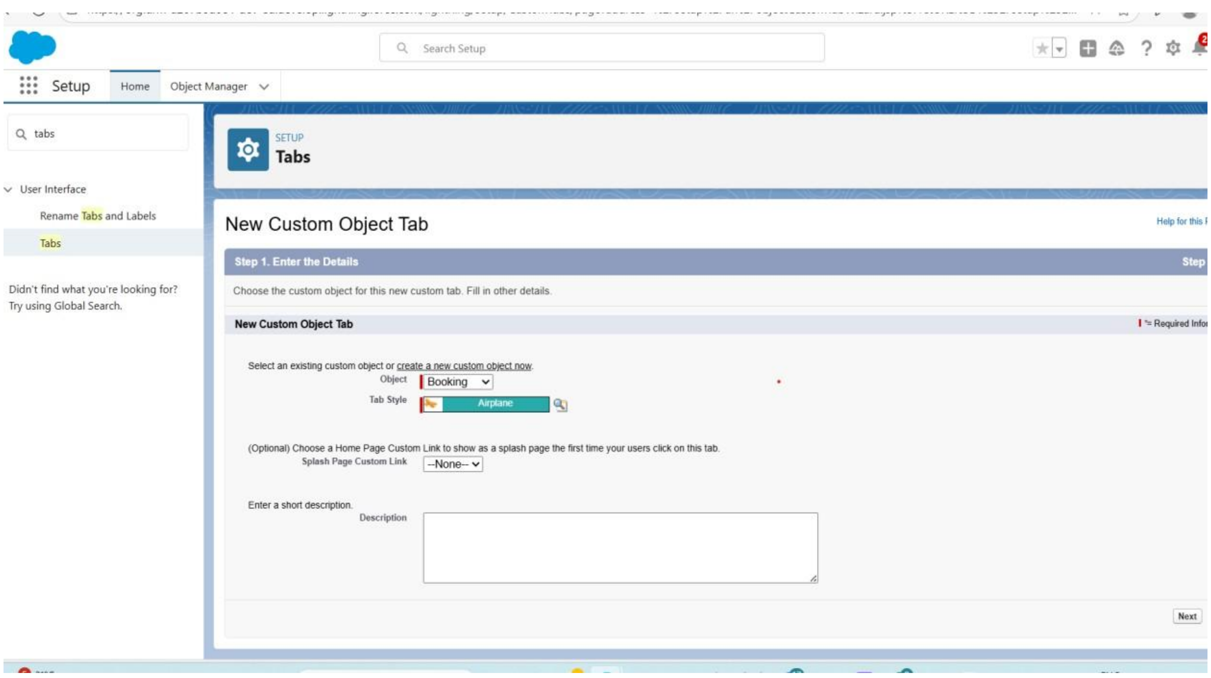


SETUP
Tabs

Step 3. Add to Custom Apps — Step 3 of 3

Choose the custom apps for which the new custom tab will be available. You may also examine or alter the visibility of tabs from the detail and edit pages of each Custom App.

Custom App	✓ Include Tab
Platform (standard__Platform)	✓
Sales (standard__Sales)	✓
Service (standard__Service)	✓
Marketing CRM Classic (standard__Marketing)	✓
Sample Console (standard__ServiceConsole)	✓
High Volume Customer Portal User	✓
Authenticated Website User	✓
App Launcher (standard__AppLauncher)	✓
Community (standard__Community)	✓
Site.com (standard__Sites)	✓
Salesforce Chatter (standard__Chatter)	✓
Content (standard__Content)	✓
Analytics Studio (standard__Insights)	✓
Sales Console (standard__LightningSalesConsole)	✓



New Custom Object Tab

Step 1. Enter the Details

Choose the custom object for this new custom tab. Fill in other details.

New Custom Object Tab

Select an existing custom object or create a new custom object now.

Object: Booking

Tab Style: Airplane

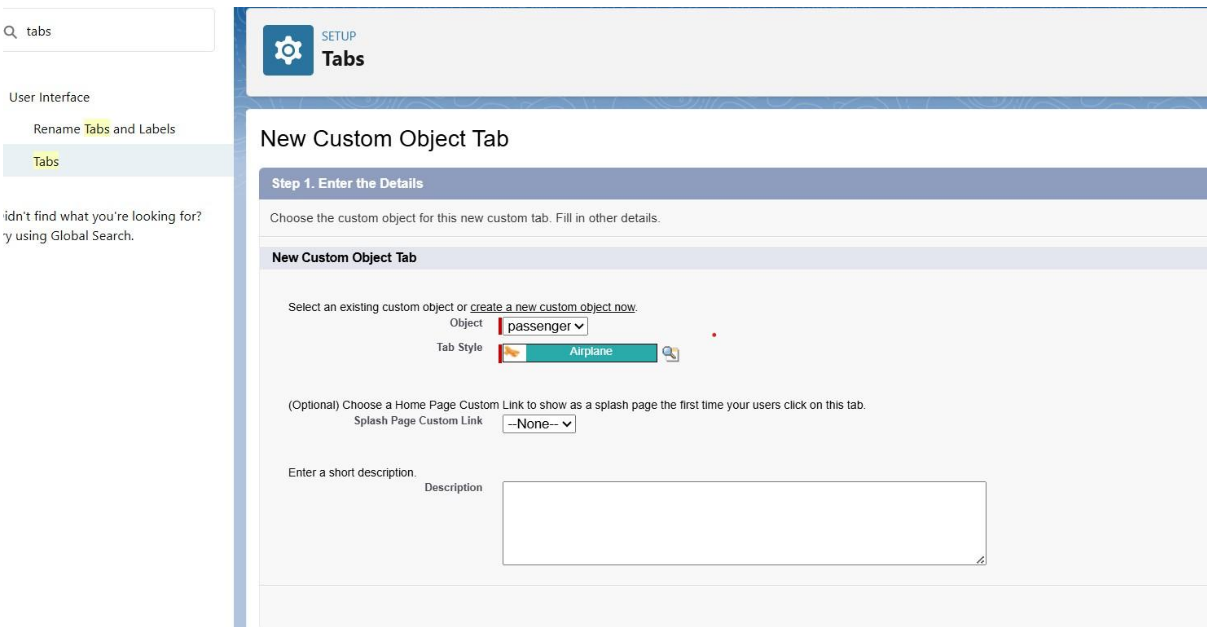
(Optional) Choose a Home Page Custom Link to show as a splash page the first time your users click on this tab.

Splash Page Custom Link: --None--

Enter a short description.

Description:

Next



New Custom Object Tab

Step 1. Enter the Details

Choose the custom object for this new custom tab. Fill in other details.

New Custom Object Tab

Select an existing custom object or create a new custom object now.

Object: passenger

Tab Style: Airplane

(Optional) Choose a Home Page Custom Link to show as a splash page the first time your users click on this tab.

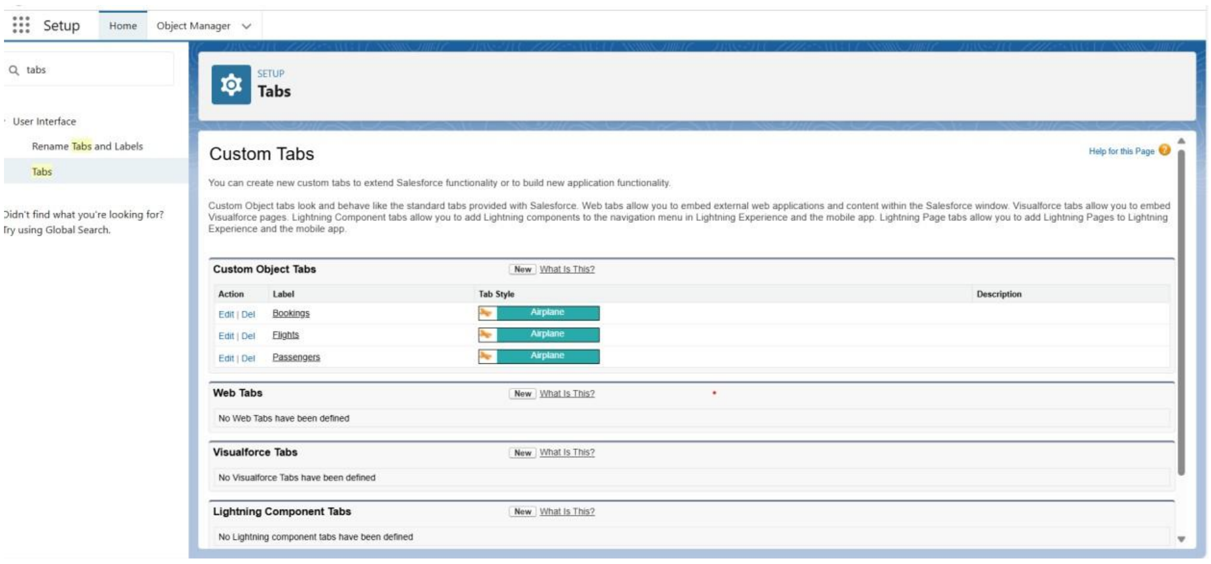
Splash Page Custom Link: --None--

Enter a short description.

Description:

Step 3: Build Lightning Record Pages

- Setup → **Lightning App Builder** → **New Page** → **Record Page**.
- Select Flight__c → design layout (Record Details + Related Lists).
- Save (don't forget to **Activate** later).



Custom Tabs

Help for this Page

You can create new custom tabs to extend Salesforce functionality or to build new application functionality.

Custom Object tabs look and behave like the standard tabs provided with Salesforce. Web tabs allow you to embed external web applications and content within the Salesforce window. Visualforce tabs allow you to embed Visualforce pages. Lightning Component tabs allow you to add Lightning components to the navigation menu in Lightning Experience and the mobile app. Lightning Page tabs allow you to add Lightning Pages to Lightning Experience and the mobile app.

Custom Object Tabs — New — What Is This?

Action	Label	Tab Style	Description
Edit | Del	Bookings	Airplane	
Edit | Del	Flights	Airplane	
Edit | Del	Passengers	Airplane	

Web Tabs — New — What Is This?

No Web Tabs have been defined

Visualforce Tabs — New — What Is This?

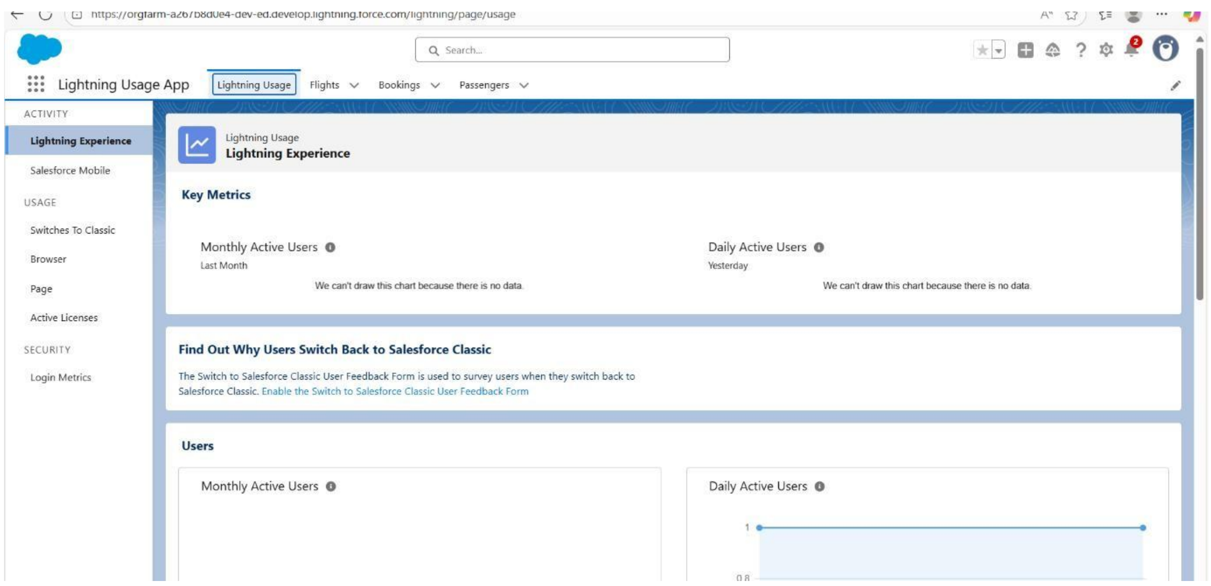
No Visualforce Tabs have been defined

Lightning Component Tabs — New — What Is This?

No Lightning component tabs have been defined

Step 4: Customize Home Page & Utility Bar

- Setup → Lightning App Builder → New → Home Page.
- Add components (Reports, Dashboard, News).
- In App Manager → Edit App → Utility Bar → add *Notes*, *Recent Items*, or custom LWC**.**



We can't draw this chart because there is no data.

We can't draw this chart because there is no data.

Find Out Why Users Switch Back to Salesforce Classic

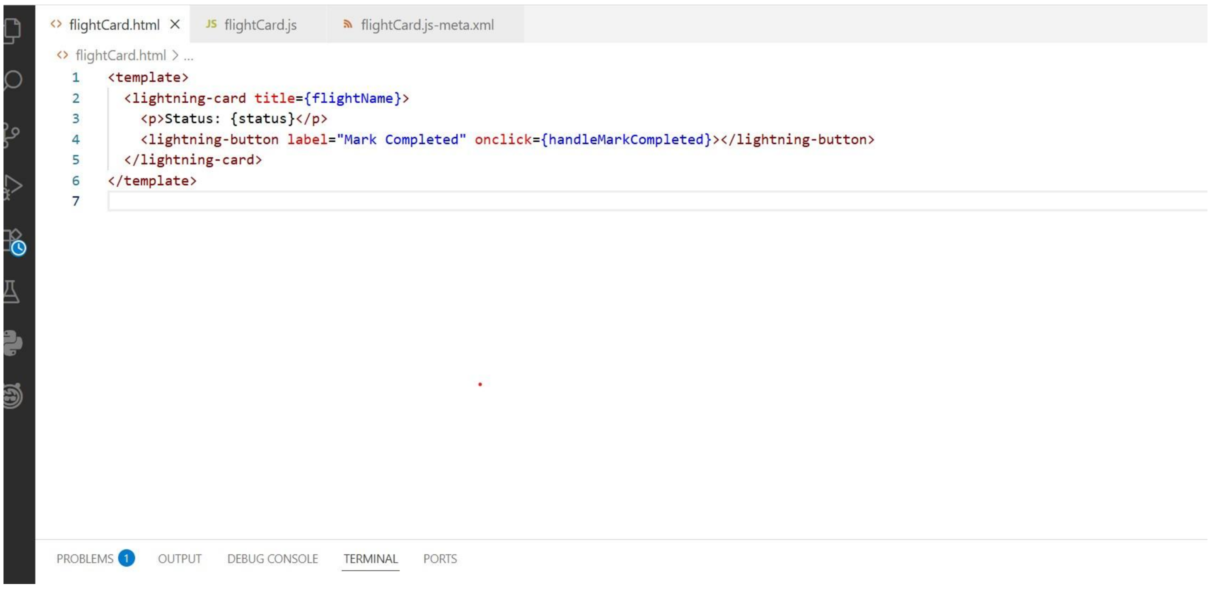
The Switch to Salesforce Classic User Feedback Form is used to survey users when they switch back to Salesforce Classic. Enable the Switch to Salesforce Classic User Feedback Form

Step 5: Create LWC Component

- In VS Code (SFDX Project): sfdx force:lightning:component:create --type lwc --componentname flightCard -outputdir force-app/main/default/lwc
- Files created: flightCard.html, flightCard.js, flightCard.js-meta.xml.

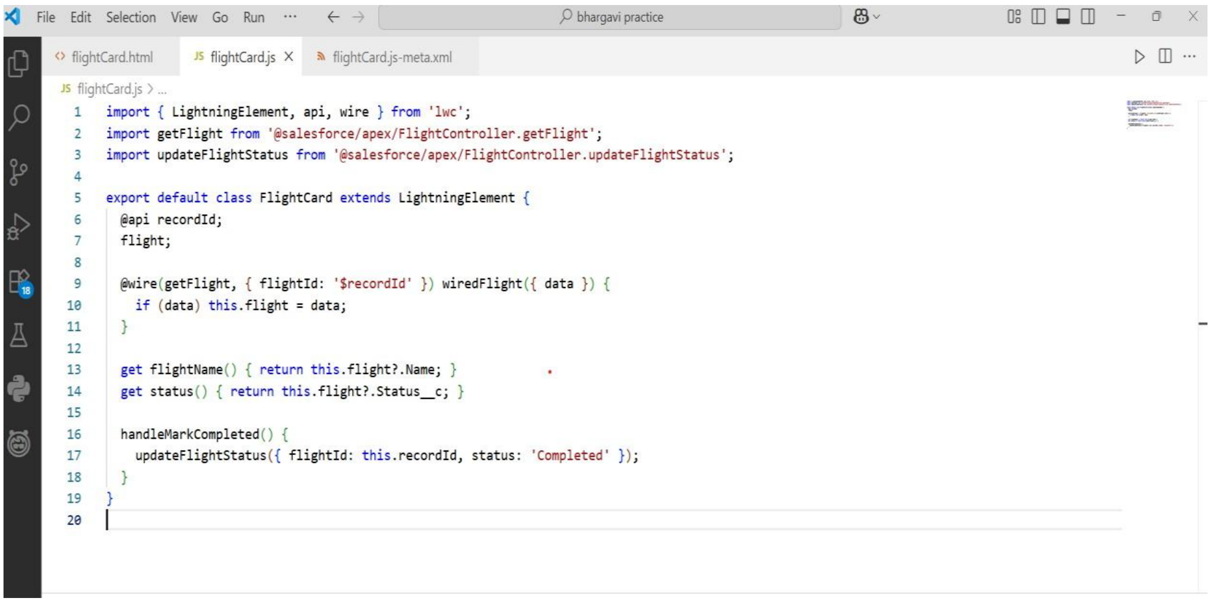
Step 6: Write LWC Code

- flightCard.html



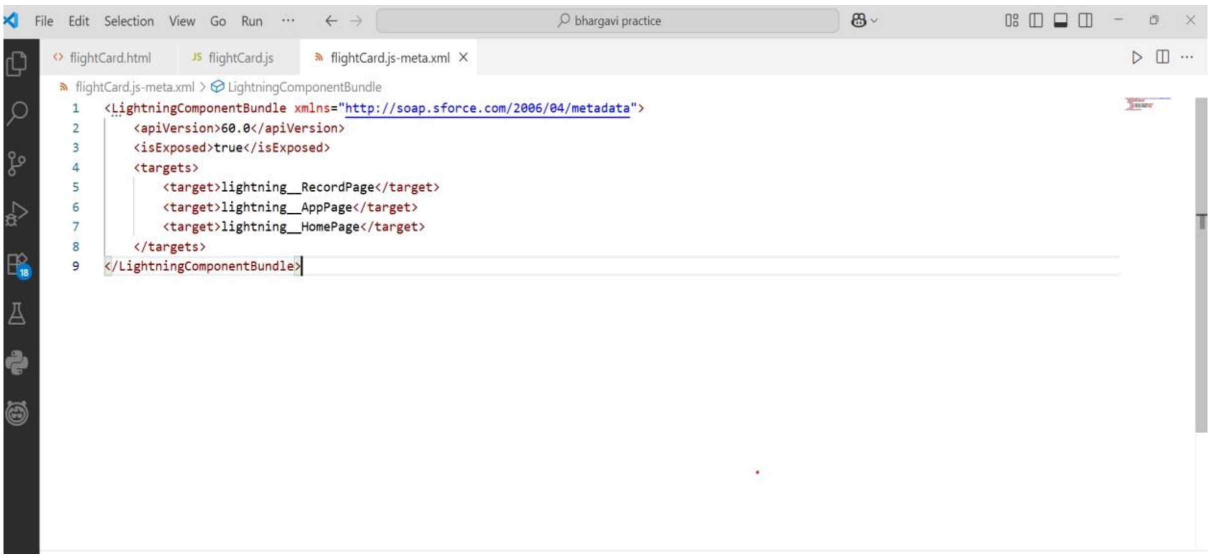
```
<template>
  <lightning-card title={flightName}>
    <p>Status: {status}</p>
    <lightning-button label="Mark Completed" onclick={handleMarkCompleted}></lightning-button>
  </lightning-card>
</template>
```

- **flightCard.js**



```
import { LightningElement, api, wire } from 'lwc';
import getFlight from '@salesforce/apex/FlightController.getFlight';
import updateFlightStatus from '@salesforce/apex/FlightController.updateFlightStatus';

export default class FlightCard extends LightningElement {
  @api recordId;
  flight;

  @wire(getFlight, { flightId: '$recordId' }) wiredFlight({ data }) {
    if (data) this.flight = data;
  }

  get flightName() { return this.flight?.Name; }
  get status() { return this.flight?.Status__c; }

  handleMarkCompleted() {
    updateFlightStatus({ flightId: this.recordId, status: 'Completed' });
  }
}
```

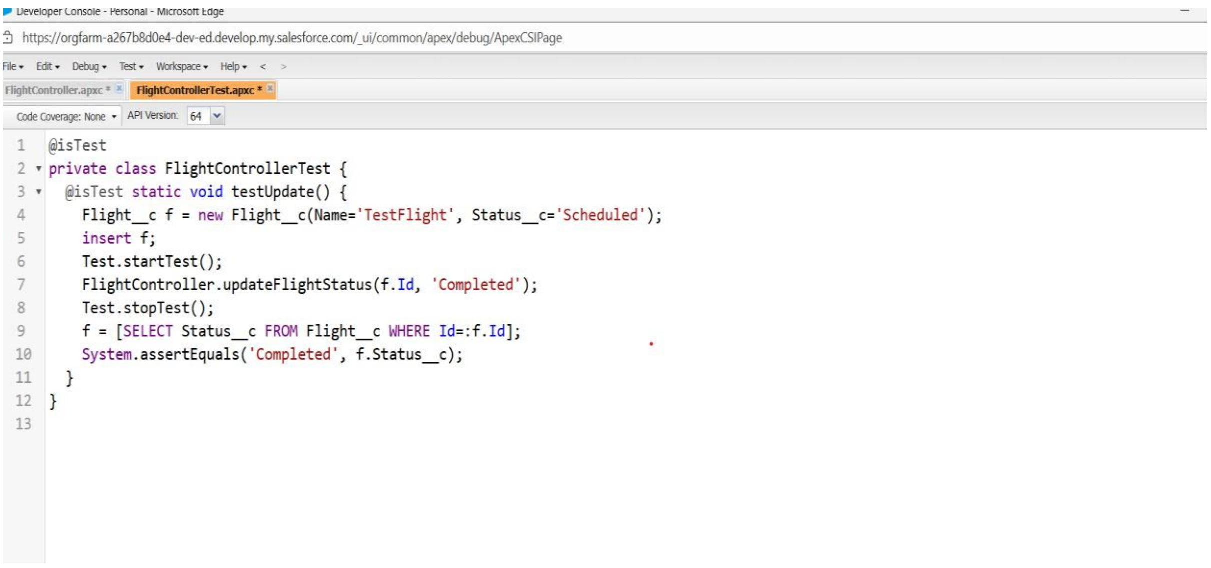
- **FlightCard.js-meta.xml**


File Edit Selection View Go Run ···
bhargavi practice
flightCard.html
flightCard.js
flightCard.js-meta.xml
flightCard.js-meta.xml > LightningComponentBundle

```xml
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>60.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__RecordPage</target>
        <target>lightning__AppPage</target>
        <target>lightning__HomePage</target>
    </targets>
</LightningComponentBundle>
```

Step 7: Create Apex Controller

- **Setup → Apex Classes → New.**


Developer Console - Personal - Microsoft Edge
https://orgfarm-a267b8d0e4-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help < >
FlightController.apxc *
FlightControllerTest.apxc *
Code Coverage: None
API Version: 64

```java
@isTest
private class FlightControllerTest {
  @isTest static void testUpdate() {
    Flight__c f = new Flight__c(Name='TestFlight', Status__c='Scheduled');
    insert f;
    Test.startTest();
    FlightController.updateFlightStatus(f.Id, 'Completed');
    Test.stopTest();
    f = [SELECT Status__c FROM Flight__c WHERE Id=:f.Id];
    System.assertEquals('Completed', f.Status__c);
  }
}
```

Step 8: Deploy LWC & Apex

- In VS Code: right-click component → SFDX: Deploy Source to Org.
- Deploy Apex class too.

Step 9: Add LWC to Lightning Page

- Setup → **Lightning App Builder** → Open Flight_Record_Page_Custom.
- Drag flightCard component onto the page.
- **Save & Activate** → assign to App, Record Type, Profile.

Step 10: Test in Salesforce App

- Open a Flight__c record.
- Verify component displays flight info and button updates status.