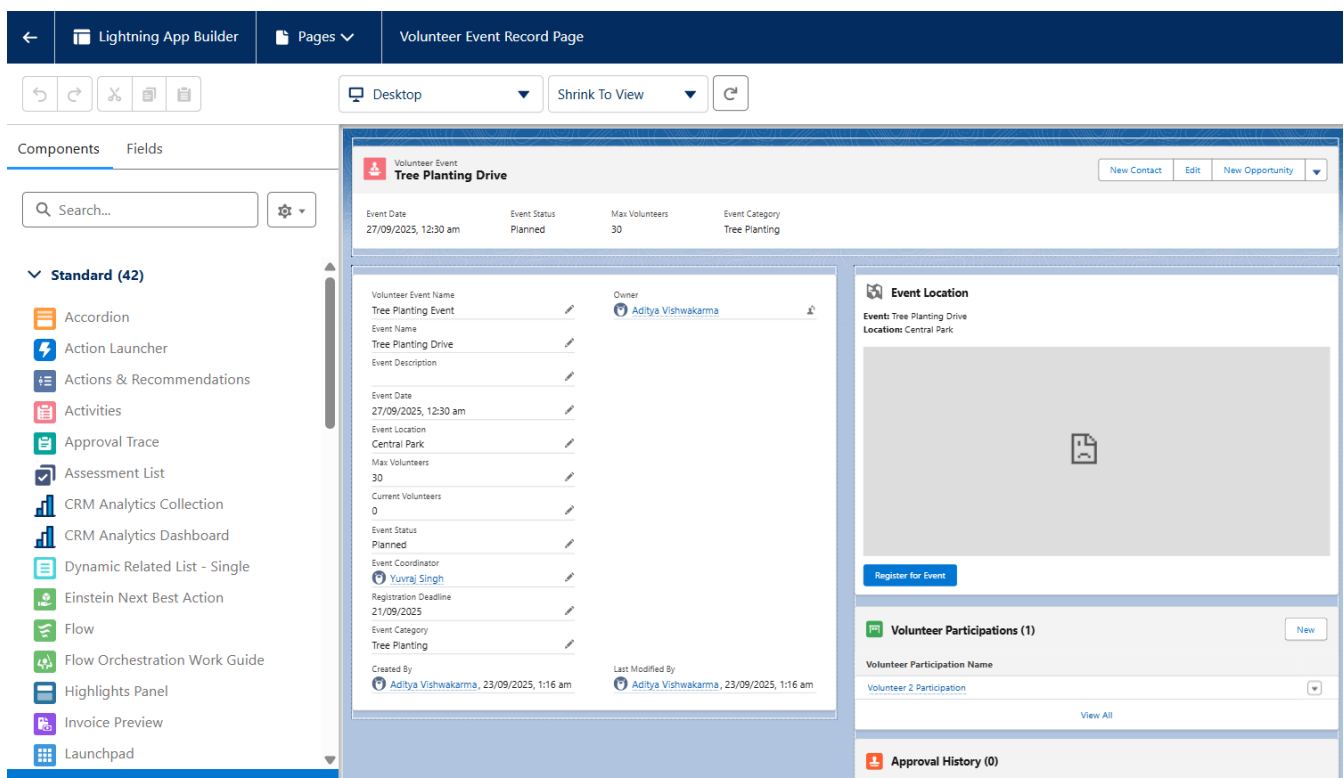


Phase 6 User Interface Development

»» Lightning App Builder & Record Pages

- Custom Volunteer Event Record Page
- In Setup → Lightning App Builder, click New → Record Page
 - ✓ Label: Volunteer Event Record Page
 - ✓ Object: Volunteer_Event__c
- Drag standard Highlights Panel and Related Lists onto the canvas.
- Add your custom Lightning Web Component (see Section 3) into a full-width region.
- Activate for the Org Default and assign to desktop and mobile.



The screenshot displays the Lightning App Builder interface for a 'Volunteer Event Record Page'. The top navigation bar includes 'Lightning App Builder', 'Pages', and the page title 'Volunteer Event Record Page'. Below the navigation bar, there are icons for undo, redo, delete, and duplicate, along with a 'Desktop' view selector and a 'Shrink To View' button. The left sidebar shows the 'Components' panel with a search bar and a list of standard components under the 'Standard (42)' category, including Accordion, Action Launcher, Actions & Recommendations, Activities, Approval Trace, Assessment List, CRM Analytics Collection, CRM Analytics Dashboard, Dynamic Related List - Single, Einstein Next Best Action, Flow, Flow Orchestration Work Guide, Highlights Panel, Invoice Preview, and Launchpad. The main canvas displays the record page layout for a 'Volunteer Event Tree Planting Drive'. The layout includes a header with 'New Contact', 'Edit', and 'New Opportunity' buttons. Below the header, there is a table with columns for Event Date, Event Status, Max Volunteers, and Event Category. The main content area is divided into two columns. The left column contains a 'Volunteer Event' section with fields for Event Name, Event Description, Event Date, Event Location, Max Volunteers, Current Volunteers, Event Status, Event Coordinator, Registration Deadline, and Event Category. The right column contains an 'Event Location' section with a map placeholder and a 'Register for Event' button. Below the map, there is a 'Volunteer Participations (1)' section with a 'New' button and a list of participation records. At the bottom, there is an 'Approval History (0)' section.

»» Tabs & Utility Bar

- Volunteer Dashboard Tab
- In Setup → Tabs, create a Lightning Page Tab
 - ✓ Label: Volunteer Dashboard

- ✓ Lightning Page: a new app page built below
- Utility Bar Component
- In Setup → App Manager, edit the Volunteer App
- Under Utility Bar, click Add, choose your LWC (e.g., Registration Modal)
- Set Label: Quick Register, Icon: user_add

Welcome to Volunteer Dashboard!

Sample Flow Report: Screen Flows

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

[View Report](#)

As of Today at 6:51 pm

Standard.RecentItems (0)

Quick Volunteer Registration

* Select Event

Select an Option

* Your Name

* Email Address

Cancel

Register

»» Lightning Web Components (LWC)


Component: volunteerEventMap

- Purpose: Display event location on Google Map
- Files:
 - ✓ volunteerEventMap.html
 - ✓ volunteerEventMap.js
 - ✓ volunteerEventMap.js-meta.xml

Component: volunteerRegistrationModal

- Purpose: Let users register for an event in a modal
- Files:
 - ✓ volunteerRegistrationModal.html

- ✓ volunteerRegistrationModal.js
- ✓ volunteerRegistrationModal.js-meta.xml

 **Quick Volunteer Registration**

* Select Event
Select an Option ▼

* Your Name

* Email Address

»» Apex with LWC & Imperative Apex Calls

Apex Class: VolunteerRegistrationController

```
public with sharing class VolunteerRegistrationController {  
    @AuraEnabled  
    public static Id registerVolunteer(Id eventId, String participantEmail) {  
        Volunteer__Participation__c vp = new Volunteer__Participation__c(  
            Event__c = eventId,  
            Participant_Email__c = participantEmail,  
            Participation_Status__c = 'Registered'  
        );  
        insert vp;  
        return vp.Id;  
    }  
}
```

- Exposed as an imperative call from volunteerRegistrationModal.js.

```

import { LightningElement, wire } from 'lwc';
import { ShowToastEvent } from 'lightning/platformShowToastEvent';
import getAvailableEvents from '@salesforce/apex/VolunteerRegistrationController.getAvailableEvents';
import registerVolunteer from '@salesforce/apex/VolunteerRegistrationController.registerVolunteer';

export default class VolunteerRegistrationModal extends LightningElement {
    selectedEventId = '';
    participantName = '';
    participantEmail = '';
    eventOptions = [];
    isRegistering = false;
    showSpinner = false;

    @wire(getAvailableEvents)
    wiredEvents({ error, data }) {
        if (data) {
            this.eventOptions = data.map(event => ({
                label: event.Event_Name__c,
                value: event.Id
            }));
        } else if (error) {
            this.showToast('Error', 'Failed to load events', 'error');
        }
    }

    handleEventChange(event) {
        this.selectedEventId = event.detail.value;
    }
}

```

»» Events in LWC

Publish an Event After Registration

- Use a custom DOM event in volunteerRegistrationModal.js:

```

const registrationEvent = new CustomEvent('registered', {
    detail: { eventId: this.recordId }
});

this.dispatchEvent(registrationEvent);

```

- Consume in volunteerEventMap.js to refresh the map or count badge.

```

validateInputs() {
    if (!this.selectedEventId || !this.participantName || !this.participantEmail) {
        this.showToast('Error', 'Please fill in all required fields', 'error');
        return false;
    }
    return true;
}

resetForm() {
    this.selectedEventId = '';
    this.participantName = '';
    this.participantEmail = '';
}

showToast(title, message, variant) {
    const evt = new ShowToastEvent({
        title: title,
        message: message,
        variant: variant
    });
    this.dispatchEvent(evt);
}

```

»» Wire Adapters

Fetch Event Record Data

- In volunteerEventMap.js, use @wire(getRecord, { recordId: '\$recordId', fields: [EVENT_NAME_FIELD, LOCATION_FIELD] }) record;
- Display record.data.fields.Event_Location__c.value on the map.

```
@wire(getAvailableEvents)
wiredEvents({ error, data }) {
  if (data) {
    this.eventOptions = data.map(event => ({
      label: event.Event_Name__c,
      value: event.Id
    }));
  } else if (error) {
    this.showToast('Error', 'Failed to load events', 'error');
  }
}
```

»» Navigation Service

Navigate to Registration Modal via Button

- In volunteerEventMap.js, import NavigationMixin and call:

```
this[NavigationMixin.Navigate]({
  type: 'standard__navItemPage',
  attributes: { apiName: 'Volunteer_Dashboard' }
});
```