PHASE - 6

Project: AI-Powered HR & Employee Management Bot

Step 1) Create Tabs for your custom objects

- Setup → Tabs → New → Custom Object Tabs
- Create tabs for: Employee__c, Leave__c, Attendance__c, Onboarding__c, Performance__c
- Assign to HR profiles and the new app (next step)

Step 2) Create the “HR Bot” Lightning App with Utility Bar

- Setup → App Manager → New Lightning App
  - Name: HR Bot
  - Navigation: Standard Navigation
  - Utility Bar: Add “HR Assistant” (you’ll add LWC in Step 7)
  - Add Items (tabs): Employees, Leaves, Attendance, Performance, Onboarding, Reports, Dashboards
  - Assign to HR Manager, Manager, Employee Profiles

Step 3) Record Pages (Lightning App Builder)

- Employee__c record page
  - Add: Highlights Panel (Compact layout), Tabs: Details, Related
  - Related lists: Leaves, Attendance, Performance
  - Optionally add LWC “employeeSearch” as a side panel for HR users
  - Activate: Assign to App = HR Bot
- Leave__c record page
  - Add Highlights Panel (show Leave Type, Dates, Status)
  - Add Related: Employee
  - Add LWC: requestLeave as a quick action (Step 6) or component on page
  - Activate: Assign to App = HR Bot

Step 4) Home Pages

- Create 3 home pages (optional but recommended)
  - HR Home: Add hrDashboard + recent items + a dashboard
  - Manager Home: Add hrDashboard (shows Pending Approvals) + dashboard
  - Employee Home: Add hrDashboard (shows My Leave Balance) + recent items
- Activate each: Assign by App (HR Bot) and optionally by Profile

Step 5) LWC scaffolding (create folders/files)
Run in your project root:

- LWC folders:
  - hrDashboard

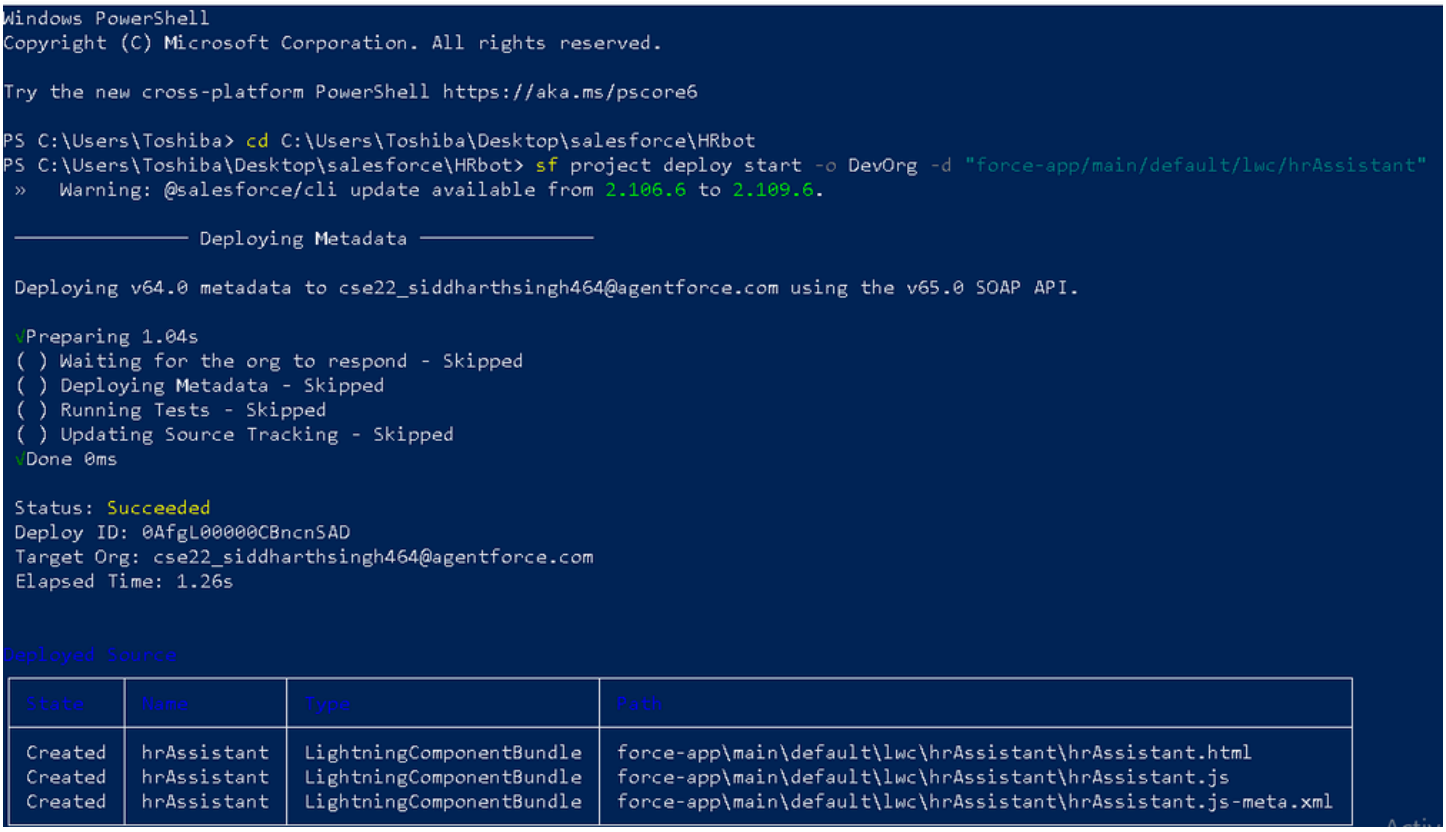
- requestLeave
- employeeSearch
- hrAssistant

Step 6) Quick Action (LWC) for Leave request

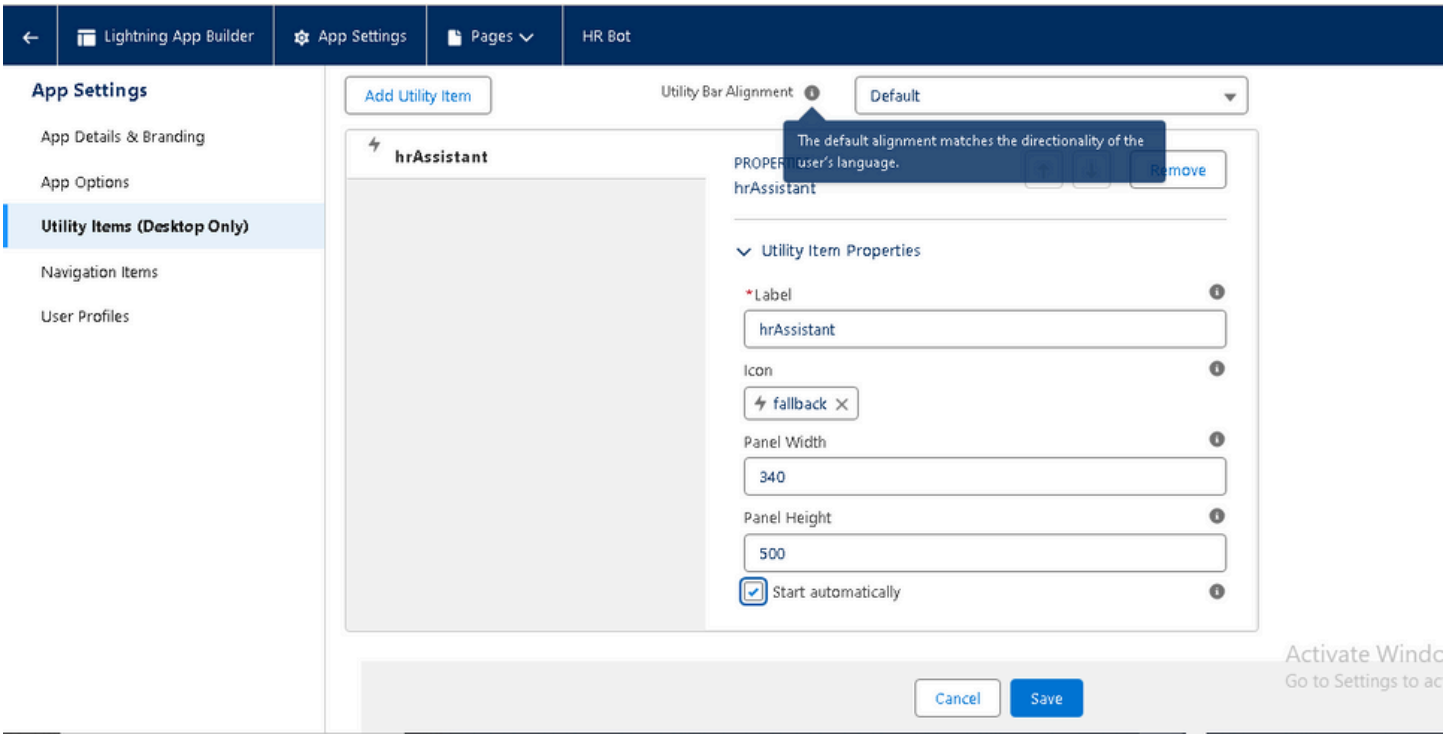
- Setup → Object Manager → Leave__c → Buttons, Links, and Actions → New Action
  - Action Type: Lightning Web Component
  - Lightning Web Component: requestLeave
  - Label: Request Leave
- Add it to the page layout:
  - Object Manager → Leave__c → Page Layouts → Salesforce Mobile and Lightning Experience Actions → drag “Request Leave” into the top section

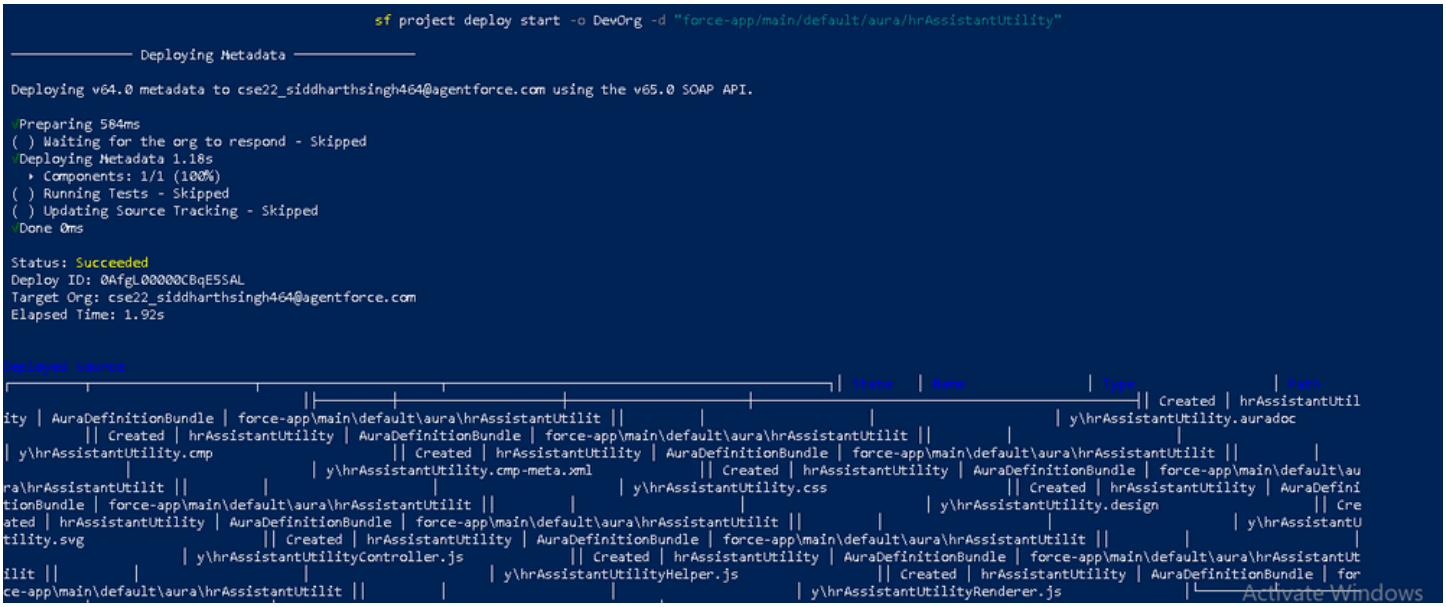
Step 7) Utility Bar LWC

- App Manager → HR Bot → Edit → Utility Items
  - Add “Open” → Choose your LWC: hrAssistant



- Label: HR Assistant → Panel Height: 500 → Start automatically: On → Save





Step 8) Apex for LWC (paste and deploy)

Create class: HR_UIController.cls

```
public with sharing class HR_UIController {
@AuraEnabled(cacheable=true)
public static Decimal getMyLeaveBalance() {
List<Employee__c> emps = [
SELECT Available_Leave_Balance__c
FROM Employee__c
WHERE User__c = :UserInfo.getUserId()
LIMIT 1
];
return emps.isEmpty() ? 0 : (emps[0].Available_Leave_Balance__c == null ? 0 :
```

```
emps[0].Available_Leave_Balance__c);
}
```

text

```
@AuraEnabled(cacheable=true)
public static Integer getMyPendingApprovals() {
    // Assumes Employee__r.Manager__c (Lookup to User) on Leave__c -> Employee__c -> Manager__c
    Integer cnt = [SELECT COUNT() FROM Leave__c WHERE Approval_Status__c = 'Submitted' AND Employee__r.Manager__c =
:UserInfo.getUserId()];
    return cnt;
}


@AuraEnabled(cacheable=true)
public static List<Attendance__c> getRecentAttendance() {
    List<Employee__c> emps = [SELECT Id FROM Employee__c WHERE User__c = :UserInfo.getUserId() LIMIT 1];
    if (emps.isEmpty()) return new List<Attendance__c>();
    return [
        SELECT Id, Attendance_Date__c
        FROM Attendance__c
        WHERE Employee__c = :emps[0].Id AND Attendance_Date__c >= :Date.today().addDays(-7)
        ORDER BY Attendance_Date__c DESC
        LIMIT 10
    ];
}


@AuraEnabled(cacheable=true)
```

```
public static Id getMyEmployeeId() {
    List<Employee__c> emps = [SELECT Id FROM Employee__c WHERE User__c = :UserInfo.getUserId() LIMIT 1];
    return emps.isEmpty() ? null : emps[0].Id;
}
}
```

Create class: HR_LeaveController.cls

```
public with sharing class HR_LeaveController {
@AuraEnabled
public static Id createLeave(Date startDate, Date endDate, String leaveType, String reason) {
if (startDate == null || endDate == null) {
throw new AuraHandledException('Start and End Date are required.');
}
List<Employee__c> emps = [SELECT Id FROM Employee__c WHERE User__c = :UserInfo.getUserId() LIMIT 1];
if (emps.isEmpty()) throw new AuraHandledException('No Employee record linked to your user.');
Leave__c l = new Leave__c(
Employee__c = emps[0].Id,
Start_Date__c = startDate,
End_Date__c = endDate,
Leave_Type__c = leaveType,
Approval_Status__c = 'Submitted'
);
try {
if (Schema.sObjectType.Leave__c.fields.contains('Reason__c') && reason != null) {
l.put('Reason__c', reason);
}
insert l;
return l.Id;
} catch (DmlException e) {
throw new AuraHandledException(e.getMessage());
}
}
}
```

Create class: HR_SearchController.cls

```
public with sharing class HR_SearchController {
@AuraEnabled(cacheable=true)
public static List<Employee__c> searchEmployees(String term) {
if (String.isBlank(term)) return new List<Employee__c>();
String q = term + '*';
List<List<SObject>> results = [FIND :q IN ALL FIELDS RETURNING Employee__c(Id, Name LIMIT 20)];
return (List<Employee__c>)results[0];
```

```
}
}
```

Create class: HR_BotController.cls

```
public with sharing class HR_BotController {
@AuraEnabled
public static String askBot(String message) {
if (String.isBlank(message)) return 'Please enter a message.';
Http h = new Http();
HttpRequest req = new HttpRequest();
req.setMethod('POST');
req.setEndpoint('callout:HR_Bot_NC/ask');
req.setHeader('Content-Type','application/json');
Map<String,Object> body = new Map<String,Object>{
'message' => message,
'userId' => UserInfo.getUserId()
};
req.setBody(JSON.serialize(body));
HttpResponse res = h.send(req);
if (res.getStatusCode() >= 200 && res.getStatusCode() < 300) {
Map<String,Object> out = (Map<String,Object>)JSON.deserializeUntyped(res.getBody());
return (String)out.get('reply');
}
throw new AuraHandledException('Bot error: ' + res.getStatus());
}
}
```

Step 9) LWC: hrDashboard (wire adapters + navigation + parent/child event)

force-app/main/default/lwc/hrDashboard/hrDashboard.html

```
<template>
<lightning-card title="HR Dashboard">
<div class="slds-p-around_medium slds-grid slds-wrap">
<div class="slds-col slds-size_1-of-1 slds-medium-size_1-of-3 slds-p-around_small">
<lightning-tile label="My Leave Balance">
<p class="slds-text-heading_large">{leaveBalance}</p>
</lightning-tile>
</div>
<div class="slds-col slds-size_1-of-1 slds-medium-size_1-of-3 slds-p-around_small">
<lightning-tile label="Pending Approvals (Manager)">
<p class="slds-text-heading_large">{pendingApprovals}</p>
<lightning-button variant="brand-outline" label="Open Leaves" onclick={openLeavesList}></lightning-
button>
</lightning-tile>
</div>
<div class="slds-col slds-size_1-of-1 slds-medium-size_1-of-3 slds-p-around_small">
<lightning-tile label="Recent Attendance">
```

```
<template if:true={attendance}>
<ul>
<template for:each={attendance} for:item="a">
<li key={a.Id}>{a.Attendance_Date__c}</li>
</template>
</ul>
</template>
</lightning-tile>
</div>
</div>
```

text

```
  <div class="slds-p-around_medium">

    <c-request-leave onleavecreated={handleLeaveCreated}></c-request-leave>

  </div>

</lightning-card>
```

```
</template>
```

force-app/main/default/lwc/hrDashboard/hrDashboard.js

```
import { LightningElement, wire } from 'lwc';
import { NavigationMixin } from 'lightning/navigation';
import { refreshApex } from '@salesforce/apex';
import getMyLeaveBalance from '@salesforce/apex/HR_UIController.getMyLeaveBalance';
import getMyPendingApprovals from '@salesforce/apex/HR_UIController.getMyPendingApprovals';
import getRecentAttendance from '@salesforce/apex/HR_UIController.getRecentAttendance';

export default class HrDashboard extends NavigationMixin(LightningElement) {
leaveBalance;
pendingApprovals;
attendance;
```

text

```
leaveBalanceWire;

pendingApprovalsWire;

attendanceWire;



@wire(getMyLeaveBalance)
```

```
wiredBalance(result) {
    this.leaveBalanceWire = result;
    if (result.data !== undefined) this.leaveBalance = result.data;
}

@wire(getMyPendingApprovals)
wiredApprovals(result) {
    this.pendingApprovalsWire = result;
    if (result.data !== undefined) this.pendingApprovals = result.data;
}

@wire(getRecentAttendance)
wiredAttendance(result) {
    this.attendanceWire = result;
    if (result.data !== undefined) this.attendance = result.data;
}

handleLeaveCreated() {
    if (this.leaveBalanceWire) refreshApex(this.leaveBalanceWire);
    if (this.pendingApprovalsWire) refreshApex(this.pendingApprovalsWire);
}

openLeavesList() {
    this[NavigationMixin.Navigate]({
```

```
    type: 'standard__objectPage',
    attributes: {
      objectApiName: 'Leave__c',
      actionName: 'list'
    }
  });
}
}
```

force-app/main/default/lwc/hrDashboard/hrDashboard.js-meta.xml

```
<?xml version="1.0" encoding="UTF-8"?> <LightningComponentBundle
xmlns="http://soap.sforce.com/2006/04/metadata"> <apiVersion>60.0</apiVersion>
<isExposed>true</isExposed> <targets> <target>lightning__HomePage</target>
<target>lightning__AppPage</target> </targets> </LightningComponentBundle>
```

Step 10) LWC: requestLeave (wire adapters + imperative Apex + event)

force-app/main/default/lwc/requestLeave/requestLeave.html

```
<template>
<lightning-card title="Request Leave">
<div class="slds-p-around_medium slds-grid slds-wrap slds-gutters">
<div class="slds-col slds-size_1-of-1 slds-medium-size_1-of-3">
<lightning-input type="date" label="Start Date" value={startDate} onchange={handleChange} data-
name="start"></lightning-input>
</div>
<div class="slds-col slds-size_1-of-1 slds-medium-size_1-of-3">
<lightning-input type="date" label="End Date" value={endDate} onchange={handleChange} data-
name="end"></lightning-input>
</div>
<div class="slds-col slds-size_1-of-1 slds-medium-size_1-of-3">
<lightning-combobox label="Leave Type" value={leaveType} options={leaveTypeOptions} onchange=
{handleTypeChange}></lightning-combobox>
</div>
<div class="slds-col slds-size_1-of-1">
<lightning-textarea label="Reason (optional)" value={reason} onchange={handleReasonChange}>
</lightning-textarea>
</div>
</div>
<div class="slds-p-around_medium">
<lightning-button variant="brand" label="Submit" onclick={submit}></lightning-button>
```

```
</div>
</lightning-card>
</template>
```

force-app/main/default/lwc/requestLeave/requestLeave.js

```
import { LightningElement, wire } from 'lwc';
import { ShowToastEvent } from 'lightning/platformShowToastEvent';
import { getObjectInfo, getPicklistValues } from 'lightning/uiObjectInfoApi';
import LEAVE_OBJECT from '@salesforce/schema/Leave__c';
import LEAVE_TYPE_FIELD from '@salesforce/schema/Leave__c.Leave_Type__c';
import createLeave from '@salesforce/apex/HR_LeaveController.createLeave';

export default class RequestLeave extends LightningElement {
startDate;
endDate;
leaveType;
reason;
```

text

```
leaveTypeOptions = [];



@wire(getObjectInfo, { objectApiName: LEAVE_OBJECT }) objectInfo;

@wire(getPicklistValues, { recordTypeId: '$objectInfo.data.defaultRecordTypeId', fieldApiName: LEAVE_TYPE_FIELD })

wiredPicklist({ data }) {

  if (data) {

    this.leaveTypeOptions = data.values.map(v => ({ label: v.label, value: v.value }));

  }

}



handleChange(e) {

  const name = e.target.dataset.name;

  if (name === 'start') this.startDate = e.target.value;

  if (name === 'end') this.endDate = e.target.value;
```

```
}

handleTypeChange(e) { this.leaveType = e.detail.value; }

handleReasonChange(e) { this.reason = e.detail.value; }



async submit() {

  try {

    const id = await createLeave({ startDate: this.startDate, endDate: this.endDate, leaveType: this.leaveType, reason: this.reason });

    this.dispatchEvent(new ShowToastEvent({ title: 'Leave Submitted', message: `Leave Id: ${id}`, variant: 'success' }));

    this.dispatchEvent(new CustomEvent('leavecreated'));

    this.startDate = this.endDate = this.leaveType = this.reason = null;

  } catch (e) {

    this.dispatchEvent(new ShowToastEvent({ title: 'Error', message: e.body ? e.body.message : e.message, variant: 'error' }));

  }

}

}
```

force-app/main/default/lwc/requestLeave/requestLeave.js-meta.xml

```
<?xml version="1.0" encoding="UTF-8"?> <LightningComponentBundle
xmlns="http://soap.sforce.com/2006/04/metadata"> <apiVersion>60.0</apiVersion>
<isExposed>true</isExposed> <targets> <target>lightning__RecordAction</target>
<target>lightning__AppPage</target> <target>lightning__HomePage</target>
<target>lightning__RecordPage</target> </targets> <targetConfigs> <targetConfig
targets="lightning__RecordAction"> <actionType>ScreenAction</actionType> </targetConfig>
</targetConfigs> </LightningComponentBundle>
```

Step 11) LWC: employeeSearch (imperative Apex + Navigation)

force-app/main/default/lwc/employeeSearch/employeeSearch.html

```
<template>
<lightning-card title="Employee Search">
<div class="slds-p-around_medium">
<lightning-input type="search" label="Search" value={term} onchange={handleTerm} placeholder="Type a
name or ID"></lightning-input>
</div>
```

```
<lightning-datatable key-field="Id" data={rows} columns={columns} hide-checkbox-column onrowaction={handleRowAction}></lightning-datatable>
</lightning-card>
</template>
```

force-app/main/default/lwc/employeeSearch/employeeSearch.js

```
import { LightningElement } from 'lwc';
import { NavigationMixin } from 'lightning/navigation';
import searchEmployees from '@salesforce/apex/HR_SearchController.searchEmployees';

export default class EmployeeSearch extends NavigationMixin(LightningElement) {
term = '';
rows = [];
columns = [
{ label: 'Name', fieldName: 'Name' },
{ type: 'button', typeAttributes: { label: 'View', name: 'view', variant: 'brand-outline' } }
];
timer;
```

text

```
handleTerm(e) {

  this.term = e.target.value;

  clearTimeout(this.timer);

  this.timer = setTimeout(() => this.search(), 300);

}



async search() {

  if (!this.term || this.term.length < 2) { this.rows = []; return; }

  this.rows = await searchEmployees({ term: this.term });

}



handleRowAction(event) {

  const recId = event.detail.row.Id;

  this[NavigationMixin.Navigate]({
```

```
      type: 'standard__recordPage',

      attributes: { recordId: recId, objectApiName: 'Employee__c', actionName: 'view' }

    });

}

}
```

force-app/main/default/lwc/employeeSearch/employeeSearch.js-meta.xml

```
<?xml version="1.0" encoding="UTF-8"?> <LightningComponentBundle
xmlns="http://soap.sforce.com/2006/04/metadata"> <apiVersion>60.0</apiVersion>
<isExposed>true</isExposed> <targets> <target>lightning__AppPage</target>
<target>lightning__HomePage</target> <target>lightning__RecordPage</target> </targets>
</LightningComponentBundle>
```

Step 12) LWC: hrAssistant (utility bar; imperative Apex callout)

force-app/main/default/lwc/hrAssistant/hrAssistant.html

```
<template>
<lightning-card title="HR Assistant">
<div class="slds-p-around_small" style="height:260px; overflow:auto;">
<template for:each={messages} for:item="m" for:index="i">
<p key={i}><strong>{m.from}:</strong> {m.text}</p>
</template>
</div>
<div class="slds-p-around_small">
<lightning-input type="text" value={draft} onchange={handleDraft} placeholder="Ask HR bot..."></lightning-
input>
<lightning-button class="slds-m-top_small" label="Send" variant="brand" onclick={send}></lightning-
button>
</div>
</lightning-card>
</template>
```

force-app/main/default/lwc/hrAssistant/hrAssistant.js

```
import { LightningElement } from 'lwc';
import askBot from '@salesforce/apex/HR_BotController.askBot';

export default class HrAssistant extends LightningElement {
messages = [];
draft = '';
```

text

```
handleDraft(e) { this.draft = e.target.value; }
```

```
async send() {
    const text = (this.draft || '').trim();
    if (!text) return;
    this.messages = [...this.messages, { from: 'Me', text }];
    this.draft = '';
    try {
      const reply = await askBot({ message: text });
      this.messages = [...this.messages, { from: 'Bot', text: reply }];
    } catch (e) {
      this.messages = [...this.messages, { from: 'Bot', text: 'Error contacting bot.' }];
    }
}
}
```

force-app/main/default/lwc/hrAssistant/hrAssistant.js-meta.xml

```
<?xml version="1.0" encoding="UTF-8"?> <LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata"> <apiVersion>60.0</apiVersion> <isExposed>true</isExposed> <targets> <target>lightning__UtilityBar</target> </targets> </LightningComponentBundle>
```

Step 13) Deploy and place components

- Deploy all:
  - sf project deploy start -o DevOrg -d force-app
- App Builder placement:
  - Home page(s): add hrDashboard
  - App page (optional): create “HR Console” app page → add hrDashboard + employeeSearch
  - Record pages: ensure Leave__c highlights and add requestLeave if desired
  - Utility Bar: confirm hrAssistant is added to the HR Bot app

Step 14) Verify against Phase 6 goals

- Lightning App Builder

    - App, Record Pages, Home Pages created and assigned
- Tabs
    - Custom object tabs visible in HR Bot app
- Utility Bar
    - HR Assistant opens and responds
- LWC + Apex
    - hrDashboard shows leave balance and pending approvals (wire to Apex)
    - requestLeave submits leave (imperative Apex) and fires CustomEvent leavecreated; hrDashboard refreshes via refreshApex
    - employeeSearch finds employees (imperative Apex + SOSL) and navigates to record
- Wire Adapters
    - requestLeave uses getObjectInfo + getPicklistValues
- Events in LWC
    - Child requestLeave -> dispatch CustomEvent('leavecreated') -> parent hrDashboard handles and refreshes
- Navigation Service
    - hrDashboard “Open Leaves” navigates to Leave__c list
    - employeeSearch row action navigates to record