

Phase 6 — User Interface Development

1. Lightning App Builder

- Configuration
 - Recruitment App with navigation items: Job Position, Candidates, Interviews, Offers Letter
- Procedure
 - App Launcher → New Lightning App → Add tabs (Jobs, Candidates, Interviews, Offers) → Assign to profiles → Save.

Screenshot

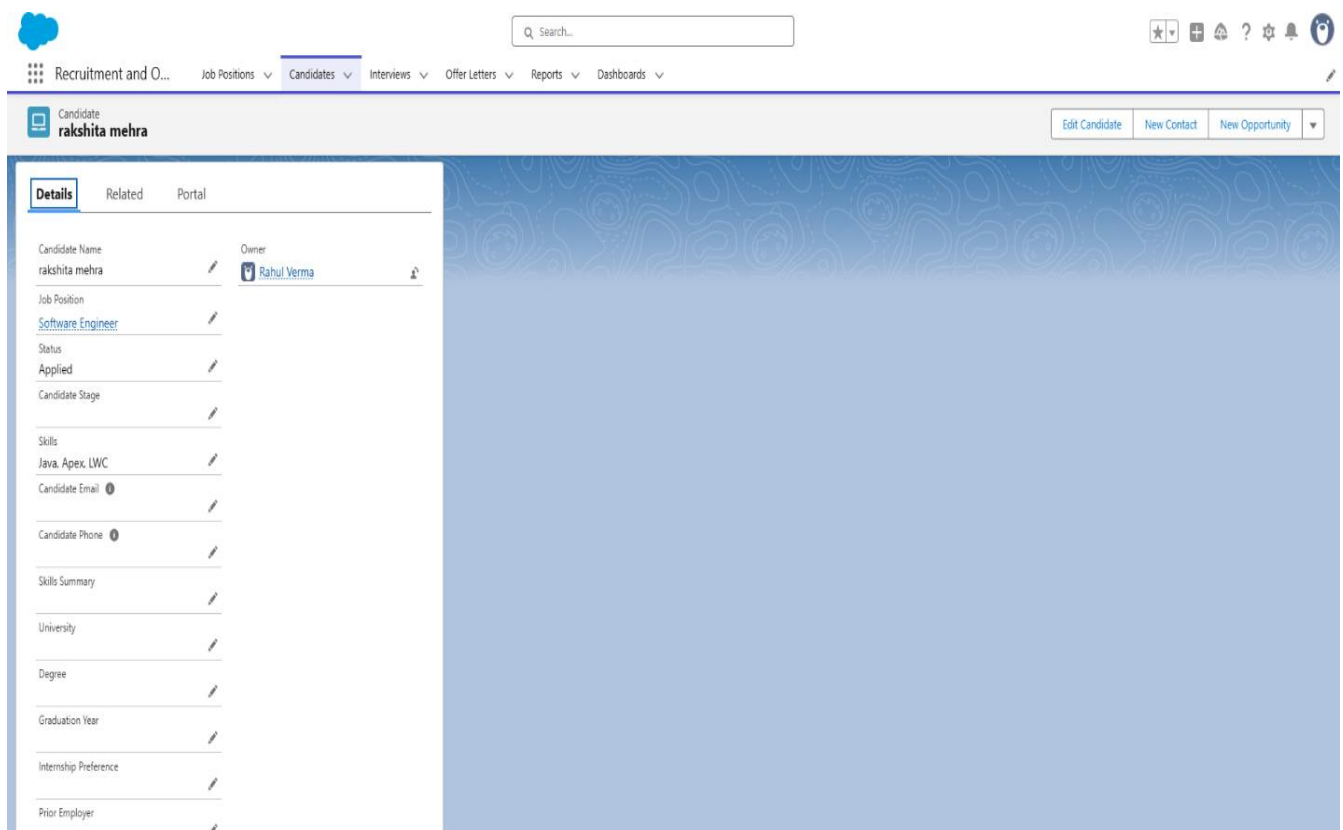
App setup page showing the app name and selected navigation items

The screenshot displays the Salesforce Lightning App Builder interface. At the top, there is a search bar and a navigation menu with items: Recruitment and O..., Job Positions, Candidates (selected), Interviews, Offer Letters, Reports, and Dashboards. The main content area is titled 'Candidates' and shows a 'Recently Viewed' section with a list of 6 items. The list includes candidate names: rakshita mehra, ocean, ramaya singh, devanshu, pranjal, and aarav. The first item, 'rakshita mehra', is highlighted. The interface also includes buttons for 'New', 'Import', 'Change Owner', and 'Assign Label'.

	Candidate Name	
1	<input type="checkbox"/> rakshita mehra	▼
2	<input type="checkbox"/> ocean	▼
3	<input type="checkbox"/> ramaya singh	▼
4	<input type="checkbox"/> devanshu	▼
5	<input type="checkbox"/> pranjal	▼
6	<input type="checkbox"/> aarav	▼

2.Record Pages

- Configuration
 - Custom record pages for Candidate and Offer with related quick actions and key related lists (Interviews, Offers).
- Procedure
 - Setup → Object Manager → Candidate/Offer → Lightning Record Pages → New → Add highlights panel, Tabs (Details, Related), and Related Lists → Activate.
- Screenshot
 - Candidate Record Page



Job Position Record Page canvas highlighting Highlights Panel and Related Lists component.

The screenshot shows the 'Job Position' record for 'DevOps Engineer'. The page has a top navigation bar with a search bar and icons. Below the navigation bar, there's a header section with 'Job Position' and 'DevOps Engineer', and buttons for 'New Contact', 'Edit', and 'New Opportunity'. A summary bar shows 'Position Status: Open', 'Department: Engineering', 'Location: Bangalore', and 'Openings: 3'. The main content area is divided into two panels. The left panel, titled 'Related', shows a list of details for the job position, including 'Job Position Name', 'Department', 'Location', 'Hiring Manager', 'Openings', 'Position Status', 'Description', 'Min Salary', and 'Role Family'. The right panel, titled 'Activity', shows a list of activities with filters for 'All time', 'All activities', and 'All types'. It also includes a section for 'Upcoming & Overdue' activities.

Job Position
DevOps Engineer

Position Status: Open | Department: Engineering | Location: Bangalore | Openings: 3

Related Details

Field	Value	Action
Job Position Name	DevOps Engineer	Edit
Owner	Poonam Rajak	
Department	Engineering	Edit
Location	Bangalore	Edit
Hiring Manager	Neha Singh	Edit
Openings	3	Edit
Position Status	Open	Edit
Description		Edit
Min Salary		Edit
Role Family		Edit

Activity

Filters: All time • All activities • All types

Refresh • Expand All • View All

Upcoming & Overdue

No activities to show.
Get started by sending an email, scheduling a task, and more.

No past activity. Past meetings and tasks marked as done show up here.

Interview Record Page canvas highlighting Highlights Panel and Related Lists component.

The screenshot shows the 'Interview' record for 'Informal Interview'. The page has a top navigation bar with a search bar and icons. Below the navigation bar, there's a header section with 'Interview' and 'Informal Interview', and buttons for 'New Contact', 'Edit', and 'New Opportunity'. A summary bar shows 'Candidate: aarav', 'Interview Date: 9/26/2025', 'Status: Completed', and 'Interviewer'. The main content area is divided into two panels. The left panel, titled 'Related', shows a list of related items, including 'Offer Letters (1)' and 'Offer Letter Name'. The right panel, titled 'Activity', shows a list of activities with filters for 'All time', 'All activities', and 'All types'. It also includes a section for 'Upcoming & Overdue' activities.

Interview
Informal Interview

Candidate: aarav | Interview Date: 9/26/2025 | Status: Completed | Interviewer

Related Details Feedback

Offer Letters (1) New

Offer Letter Name

Full time work View All

Activity

Filters: All time • All activities • All types

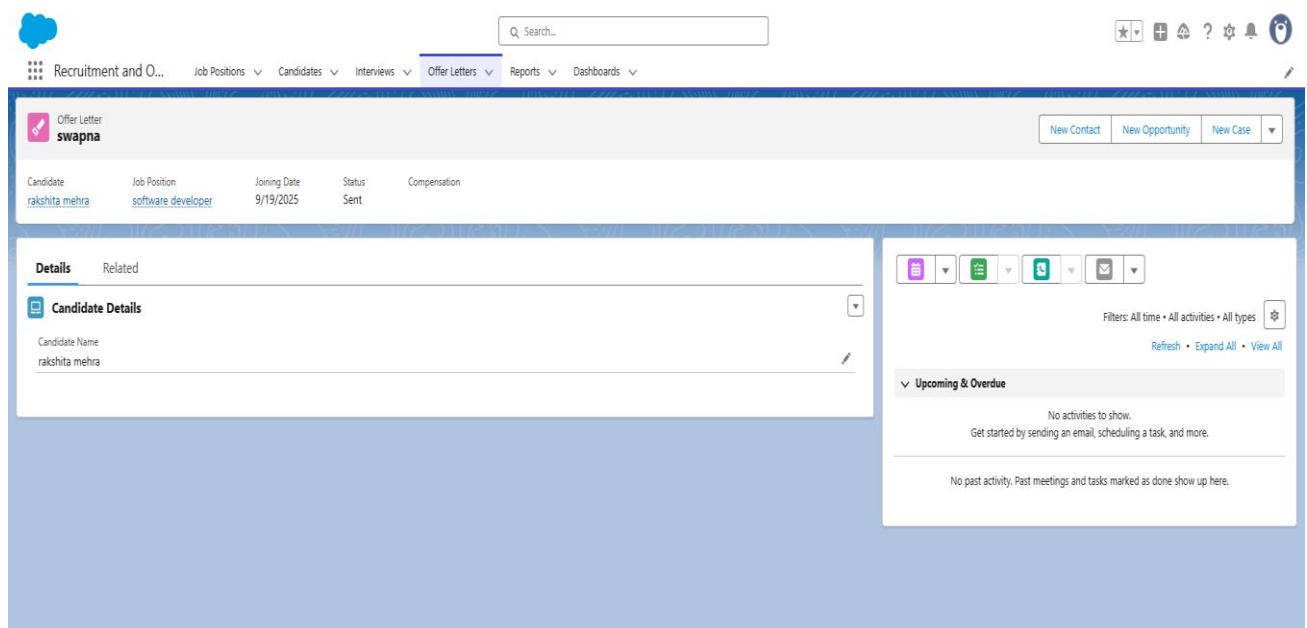
Refresh • Expand All • View All

Upcoming & Overdue

No activities to show.
Get started by sending an email, scheduling a task, and more.

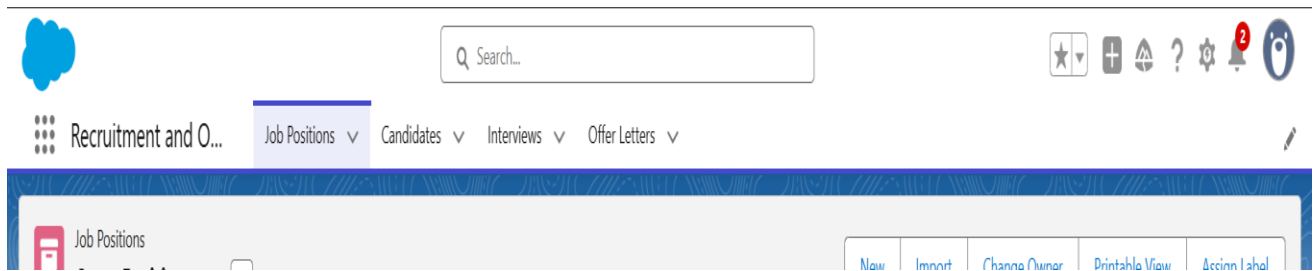
No past activity. Past meetings and tasks marked as done show up here.

Offer Letters canvas highlighting Highlights Panel and Related Lists component.



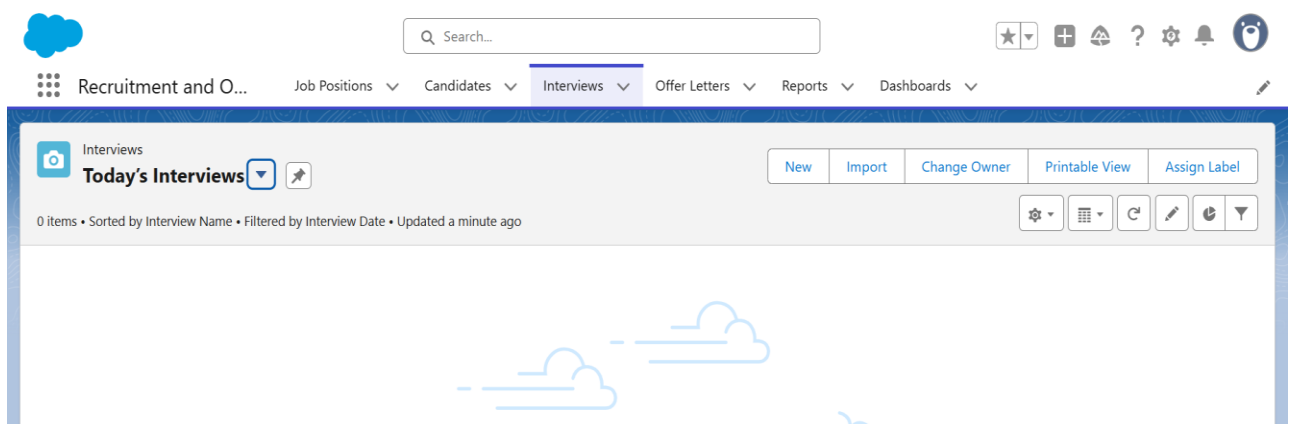
3. Tabs

- Configuration
 - Ensure standard/custom tabs exist for Jobs (Job_Position__c), Candidates (Candidate__c), Interviews (Interview__c), Offers (Offer_Letter__c).
- Procedure
 - Setup → Tabs → New (if needed) for each custom object → Add to the Recruitment App navigation.
- Screenshot
 - Tabs list showing each object's tab enabled/visible



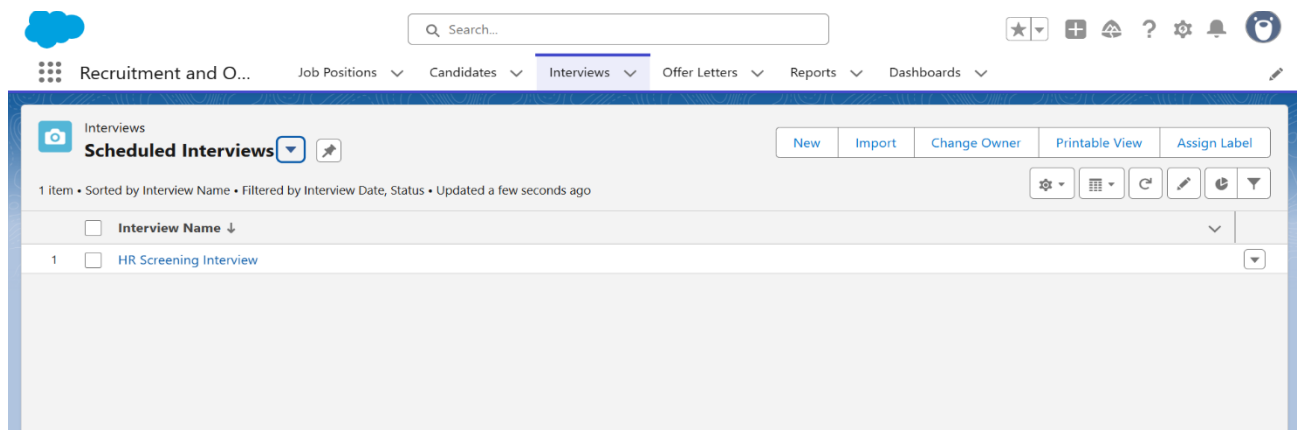
4. Home Page Layouts

- Configuration
 - Home page with “Todays Interviews” list view, Recent Items.
- Procedure
 - Setup → Lightning App Builder → Home Page → New → Add List View (Interviews with Today filter), Recent Items, and any dashboard component → Assign as Org Default or App Default.
- Screenshot
 - Home page canvas with List View component configured to “My Interviews Today.”



5. Utility Bar

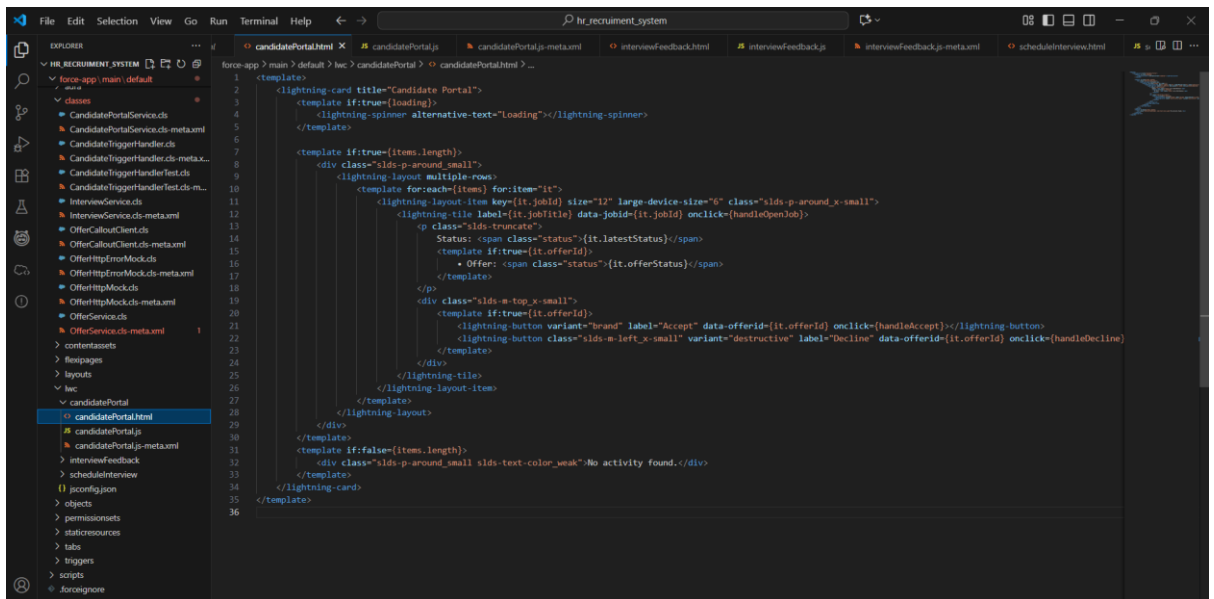
- Configuration
 - Utility Bar item “Schedule Interview” quick action or LWC for rapid scheduling.
- Procedure
 - App Manager → Edit Recruitment App → Utility Items → Add Utility Item: “Schedule Interview” → Assign component/action and icon → Save.
- Screenshot
 - App Utility Bar configuration panel showing the “Schedule Interview” item.



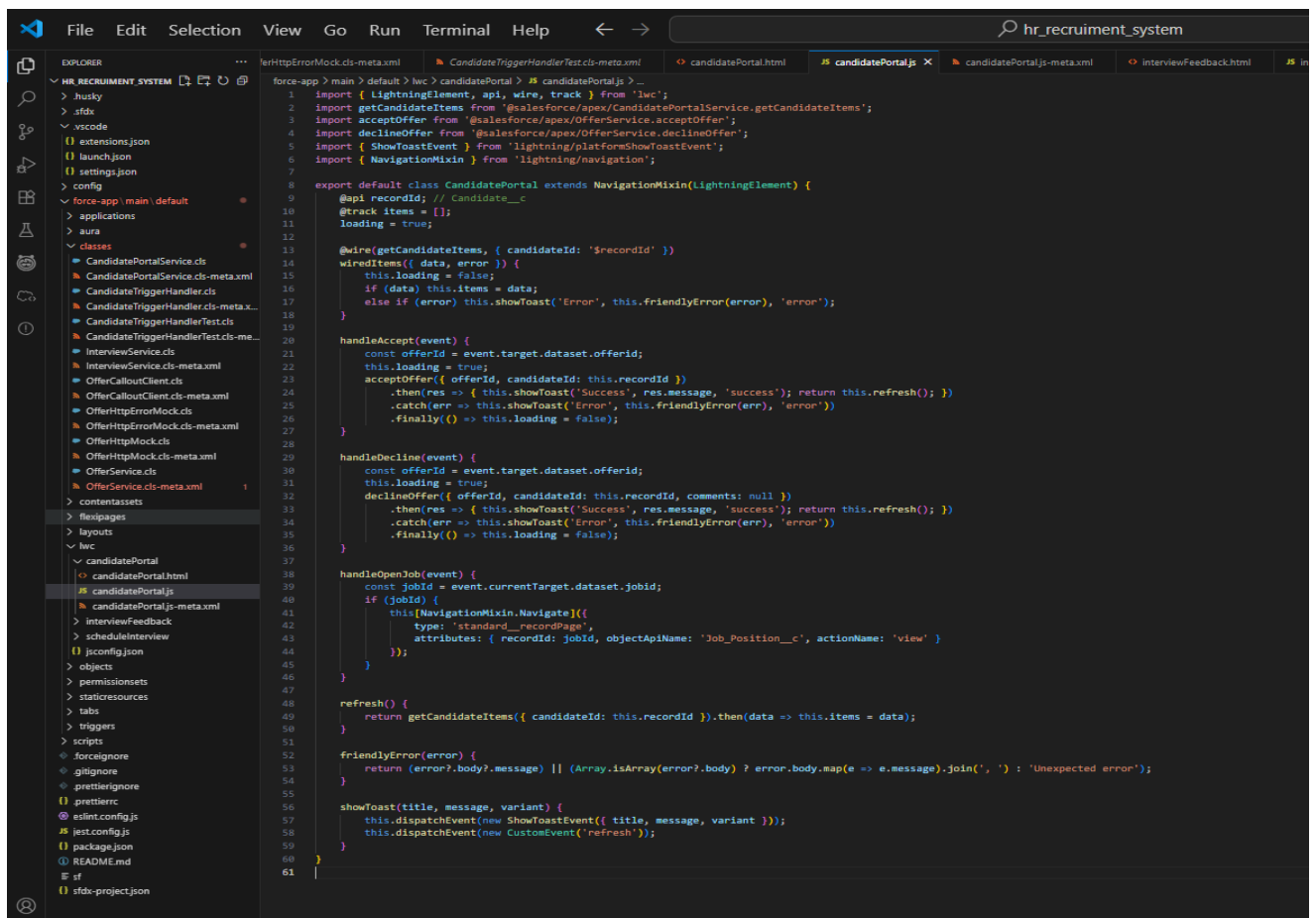
6. LWC: Candidate Portal

- Configuration
 - LWC allows candidates to view applied job, current status/offer, and Accept/Decline actions; uses wire for job list.
- Procedure
 - Create LWC candidatePortal with HTML (status/offer panel), JS using `@wire(getJobs)` to load jobs, and buttons for Accept/Decline calling imperative Apex methods in OfferService.

- Screenshot
 - candidatePortal.html

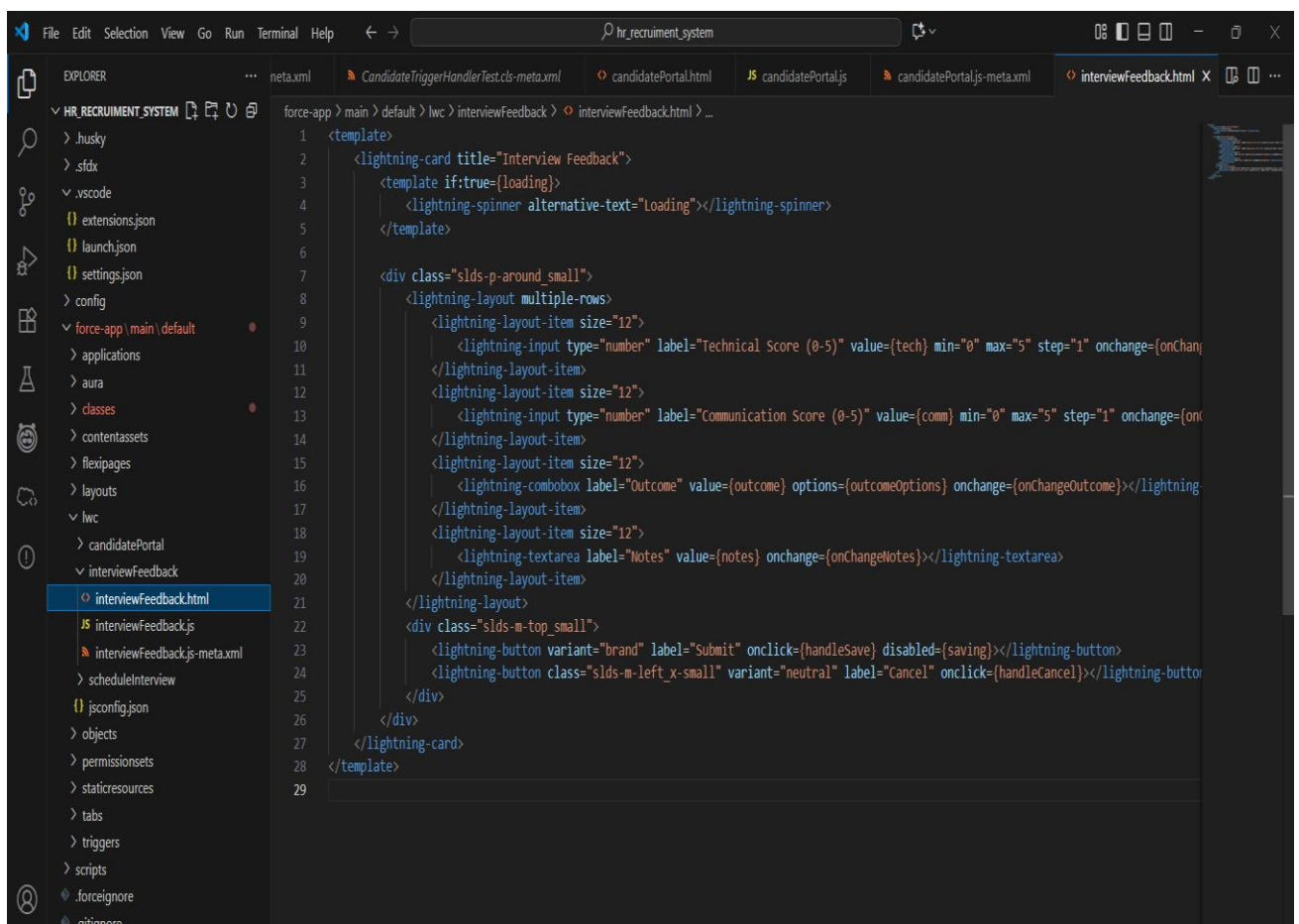


- candidatePortal.js



7. LWC: Interview Feedback

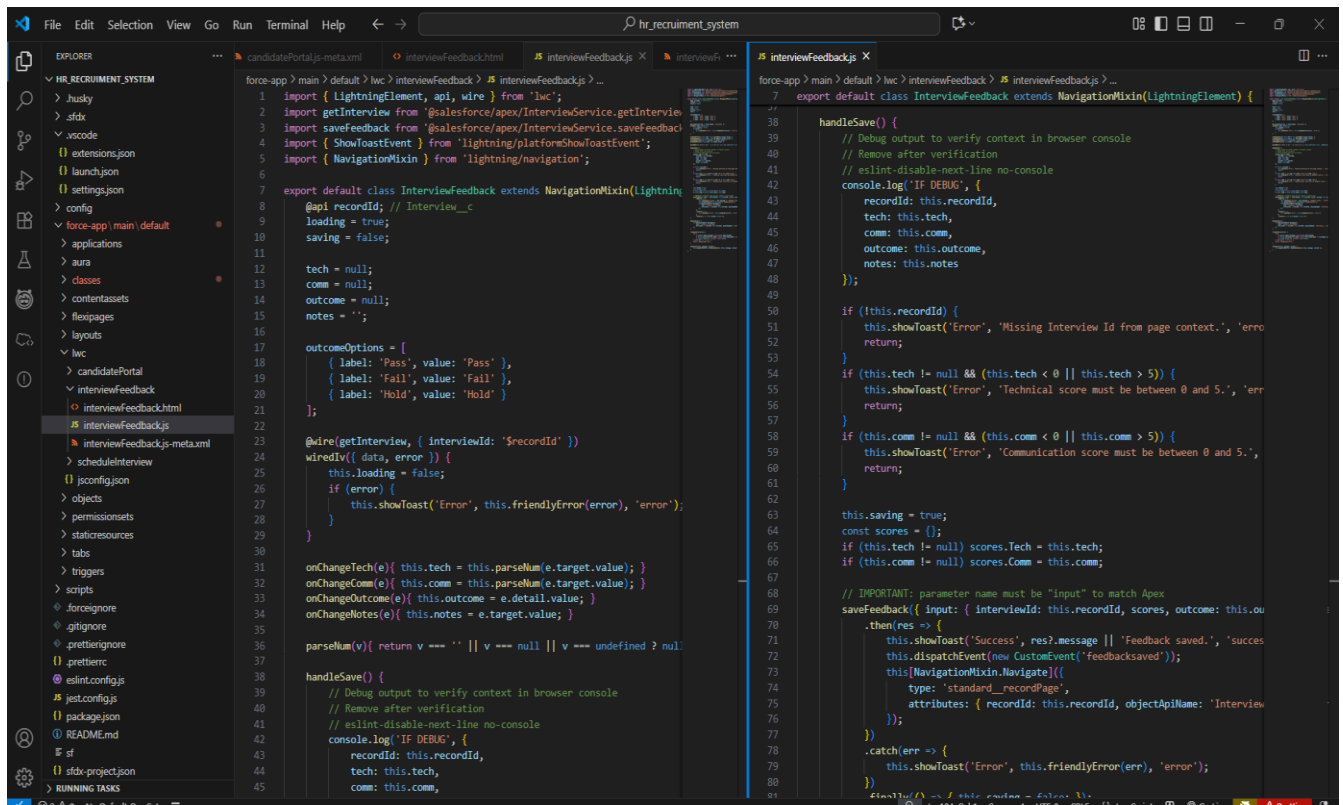
- Configuration
 - LWC for interviewers to enter scores/notes and submit feedback; uses imperative Apex to save.
- Procedure
 - Create LWC interviewFeedback with inputs (score fields, notes) and a Save button calling Apex saveFeedback(interviewId, scores, notes); show to Interviewer profile via Record Page.
- Screenshot
 - interviewFeedback.html



The screenshot shows a VS Code editor window with the file explorer on the left and the code editor on the right. The file explorer shows the project structure for 'hr_recruitment_system', with the 'interviewFeedback.html' file selected under the 'lwc' directory. The code editor displays the HTML template for the 'Interview Feedback' LWC component. The template includes a loading spinner, input fields for 'Technical Score' and 'Communication Score', a combobox for 'Outcome', a text area for 'Notes', and 'Submit' and 'Cancel' buttons. The code is as follows:

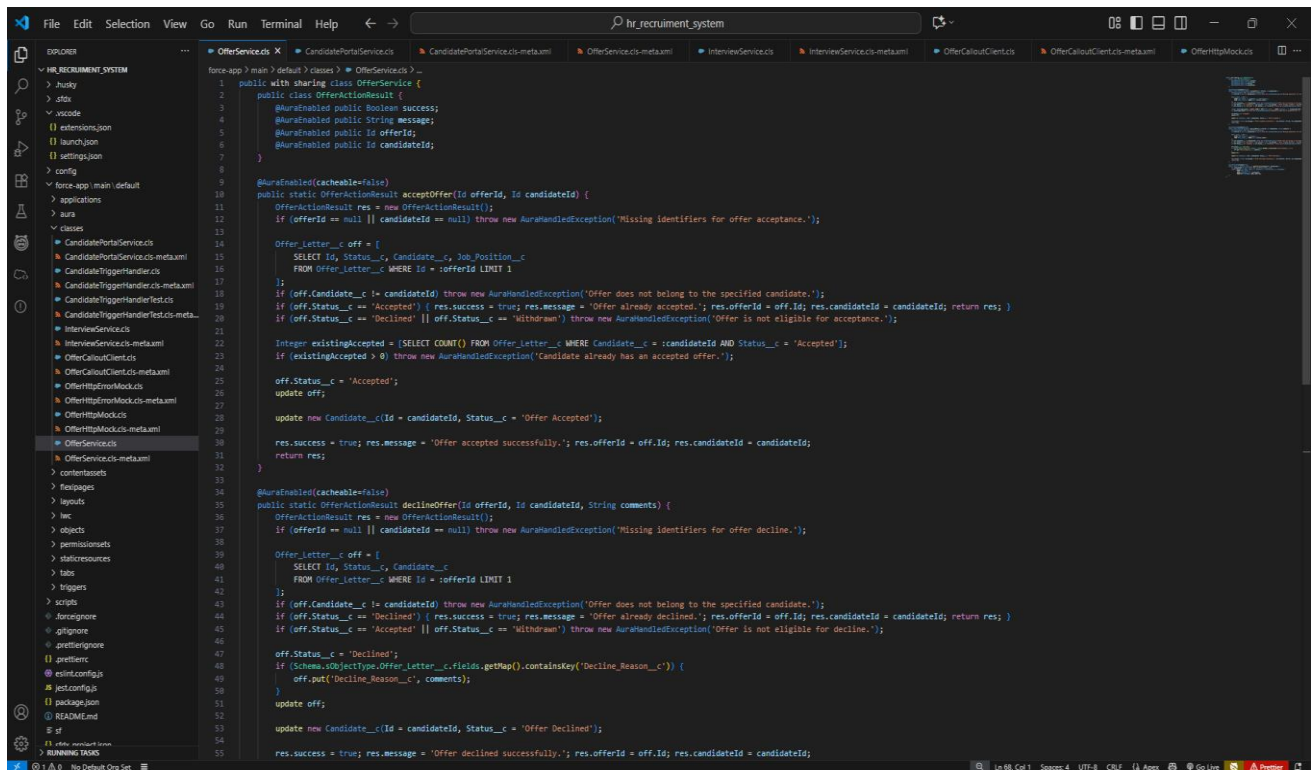
```
1 <template>
2 <lightning-card title="Interview Feedback">
3   <template if:true={loading}>
4     <lightning-spinner alternative-text="loading"></lightning-spinner>
5   </template>
6
7   <div class="slds-p-around_small">
8     <lightning-layout multiple-rows>
9       <lightning-layout-item size="12">
10        <lightning-input type="number" label="Technical Score (0-5)" value={tech} min="0" max="5" step="1" onchange={onChangeTech}></lightning-input>
11      </lightning-layout-item>
12      <lightning-layout-item size="12">
13        <lightning-input type="number" label="Communication Score (0-5)" value={comm} min="0" max="5" step="1" onchange={onChangeComm}></lightning-input>
14      </lightning-layout-item>
15      <lightning-layout-item size="12">
16        <lightning-combobox label="Outcome" value={outcome} options={outcomeOptions} onchange={onChangeOutcome}></lightning-combobox>
17      </lightning-layout-item>
18      <lightning-layout-item size="12">
19        <lightning-textarea label="Notes" value={notes} onchange={onChangeNotes}></lightning-textarea>
20      </lightning-layout-item>
21    </lightning-layout>
22    <div class="slds-m-top_small">
23      <lightning-button variant="brand" label="Submit" onclick={handleSave} disabled={saving}></lightning-button>
24      <lightning-button class="slds-m-left_x-small" variant="neutral" label="Cancel" onclick={handleCancel}></lightning-button>
25    </div>
26  </div>
27 </lightning-card>
28 </template>
29
```


- interviewFeedback.js



8. Apex with LWC

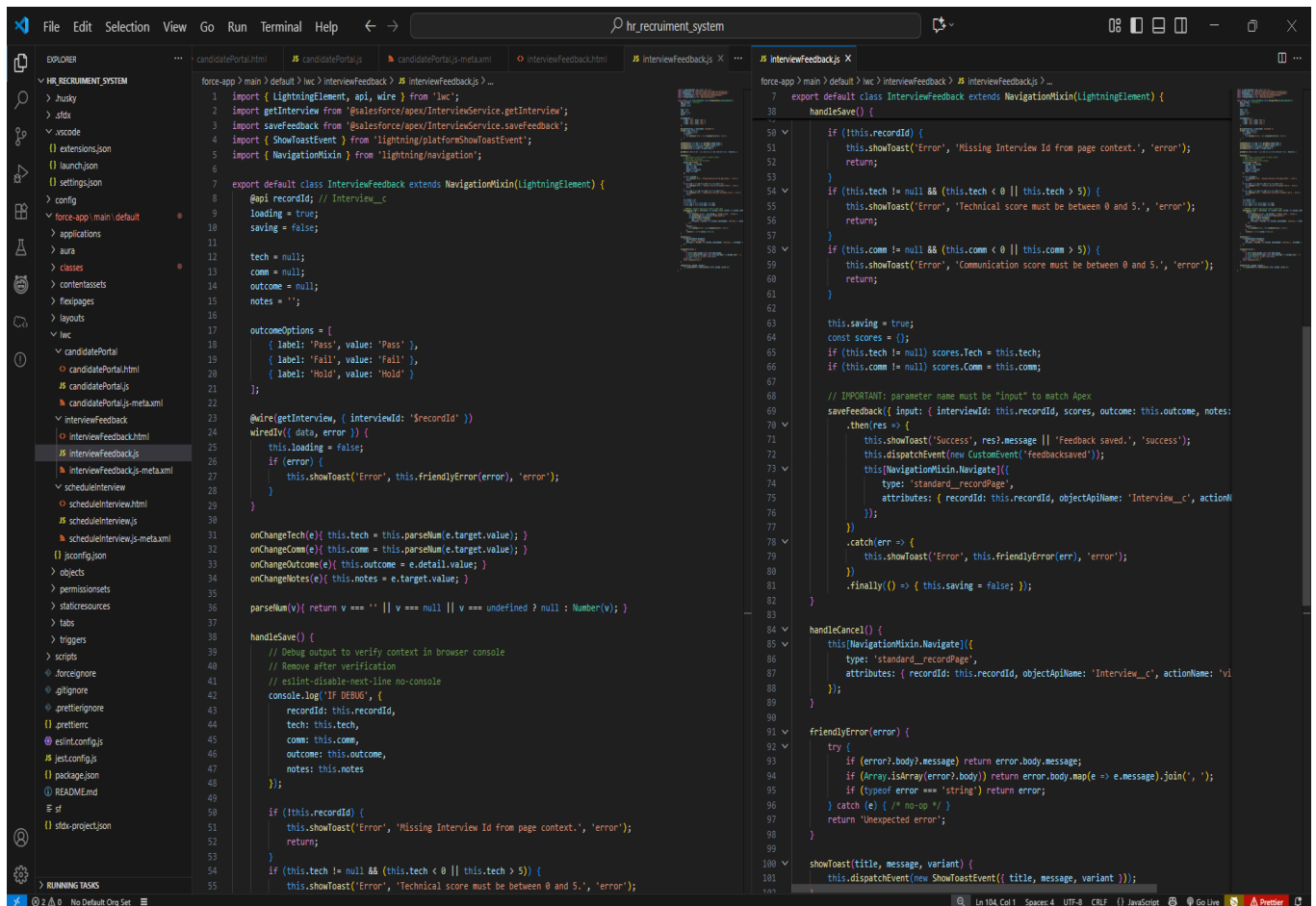
- Configuration
 - Expose @AuraEnabled(cacheable=true) for read methods (getJobs, getCandidateOffers) and non-cacheable @AuraEnabled for write actions (acceptOffer, declineOffer, saveFeedback).
- Procedure
 - Add methods in OfferService/InterviewService with with sharing; return minimal DTOs; handle exceptions with meaningful messages.
- Screenshot
 - Apex class showing @AuraEnabled methods used by both LWCs.



9. Events in LWC

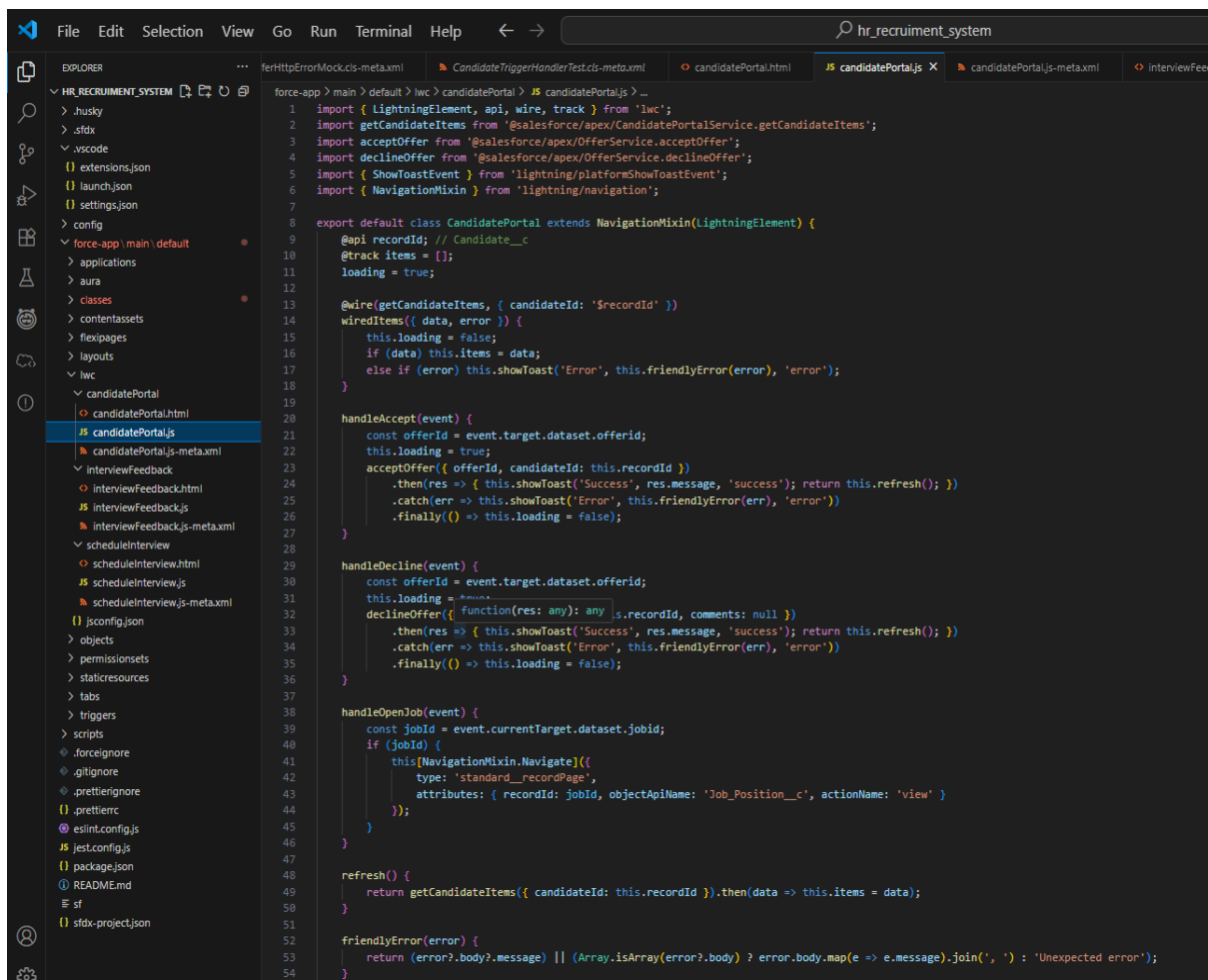
- Configuration
 - Use CustomEvent to bubble save/refresh signals back to parent or page (e.g., “feedbacksaved”, “offerupdated”).
- Procedure
 - Dispatch events after successful Apex calls; parent listens to refresh UI or navigate.

- Screenshot
- interviewFeedback.js



10.Wire Adapters

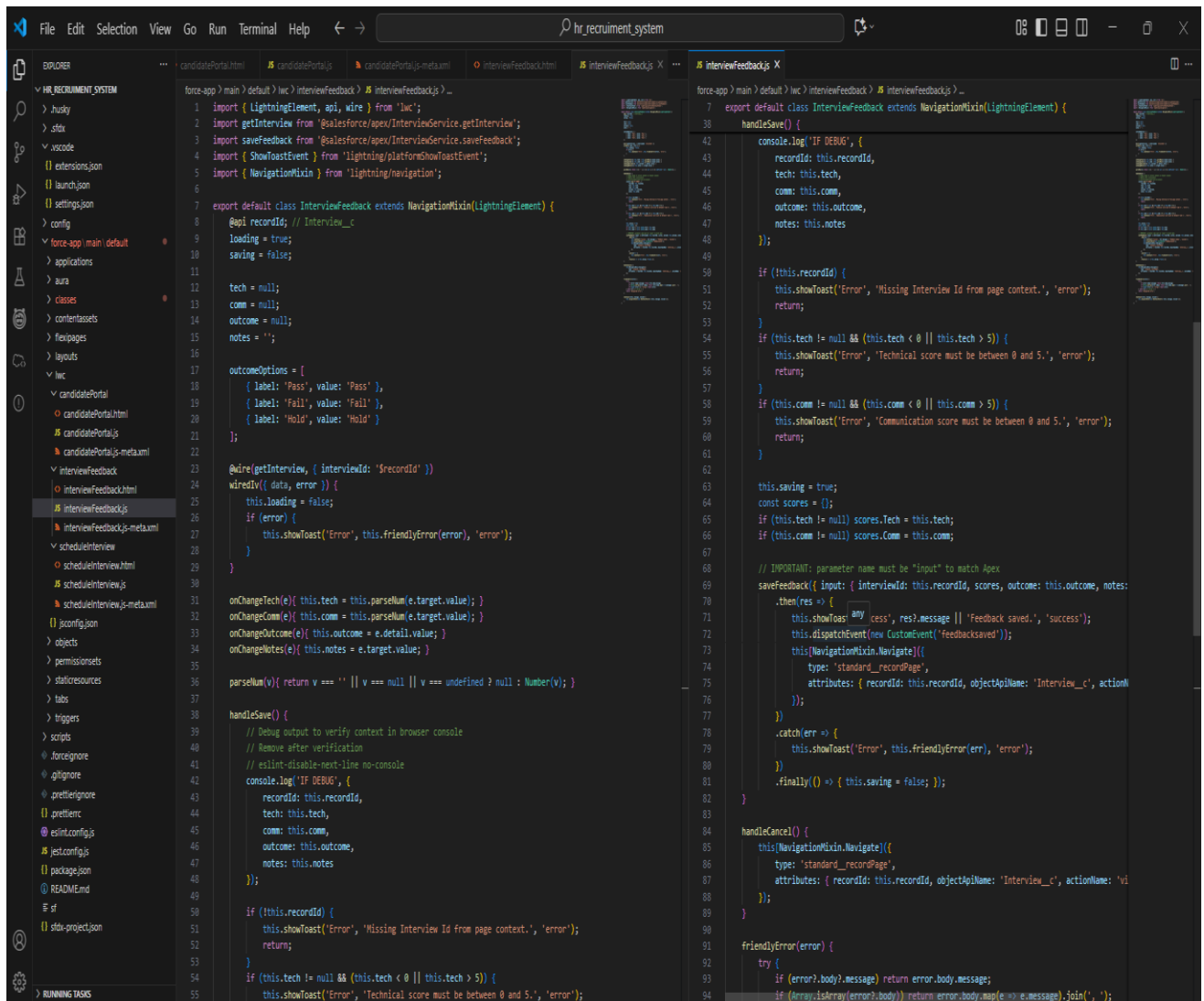
- Configuration
 - @wire for read-only, cacheable data (jobs list, candidate current status).
- Procedure
 - Implement @wire(getJobs) property form; handle error/data states; prefer getRecord/getFieldValue for simple field reads when on record pages.
- Screenshot
 - candidatePortal.js



```
1 import { LightningElement, api, wire, track } from 'lwc';
2 import getCandidateItems from '@salesforce/apex/CandidatePortalService.getCandidateItems';
3 import acceptOffer from '@salesforce/apex/OfferService.acceptOffer';
4 import declineOffer from '@salesforce/apex/OfferService.declineOffer';
5 import { ShowToastEvent } from 'lightning/platformShowToastEvent';
6 import { NavigationMixin } from 'lightning/navigation';
7
8 export default class CandidatePortal extends NavigationMixin(LightningElement) {
9   @api recordId; // Candidate__c
10   @track items = [];
11   loading = true;
12
13   @wire(getCandidateItems, { candidateId: '$recordId' })
14   wiredItems({ data, error }) {
15     this.loading = false;
16     if (data) this.items = data;
17     else if (error) this.showToast('Error', this.friendlyError(error), 'error');
18   }
19
20   handleAccept(event) {
21     const offerId = event.target.dataset.offerId;
22     this.loading = true;
23     acceptOffer({ offerId, candidateId: this.recordId })
24       .then(res => { this.showToast('Success', res.message, 'success'); return this.refresh(); })
25       .catch(err => this.showToast('Error', this.friendlyError(err), 'error'))
26       .finally(() => this.loading = false);
27   }
28
29   handleDecline(event) {
30     const offerId = event.target.dataset.offerId;
31     this.loading = true;
32     declineOffer({ offerId, candidateId: this.recordId })
33       .then(res => { this.showToast('Success', res.message, 'success'); return this.refresh(); })
34       .catch(err => this.showToast('Error', this.friendlyError(err), 'error'))
35       .finally(() => this.loading = false);
36   }
37
38   handleOpenJob(event) {
39     const jobId = event.currentTarget.dataset.jobId;
40     if (jobId) {
41       this[NavigationMixin.Navigate]({
42         type: 'standard__recordPage',
43         attributes: { recordId: jobId, objectApiName: 'Job_Position__c', actionName: 'view' }
44       });
45     }
46   }
47
48   refresh() {
49     return getCandidateItems({ candidateId: this.recordId }).then(data => this.items = data);
50   }
51
52   friendlyError(error) {
53     return (error?.body?.message) || (Array.isArray(error?.body) ? error.body.map(e => e.message).join(', ') : 'Unexpected error');
54   }
55 }
```

11. Imperative Apex Calls

- Configuration
 - Use imperative calls for writes and conditional fetches (accept/decline/saveFeedback), with UI spinners and toasts.
- Procedure
 - Call apexMethod({ params }) in async/await, try/catch; show success/error toasts and refresh data via events or refreshApex.
- Screenshot
 - interviewFeedback.js



12. Navigation Service

- Configuration
 - Navigate to Candidate/Offer records after actions (Accept/Decline, Feedback Saved).
- Procedure
 - Import NavigationMixin; call this[NavigationMixin.Navigate] with standard__recordPage targeting the relevant recordId.
- Screenshot
 - candidatePortal.js

