**Requirement 1**

**Solution :**

1. I have created a custom application and Lightning record page - [Account\_Contact\_Details](https://github.com/sb260193/Assignment-Bonial/blob/main/force-app/main/default/applications/Account_Contact_Details.app-meta.xml).
2. I have created a lightning web component - **AccountContacts**
3. We can also add SLD design to the solution, but in the interest of time, I have focused more on the logic front.
4. To test the implementation. After the installation of the package.
5. Go to the app launcher and search for Account Contact Details
6. Click to open the application, and there will be a tab - account contact details
7. Select the account - any account record.
8. Once selected, it will display contact records related to it.
9. Click on the edit button and try to edit the record.
10. On clicking the save button, it should save the details and refresh the contact details table.
11. Please find the attached screen recording, which has the demo steps:

[Question 1 solution video](https://drive.google.com/file/d/1zjjOAKEfF1DS2Zn-ieU119LBDgfhQeaS/view)

**Requirement 2**

**Solution:**

1. I have created a trigger on Opportunity Object with the name OpportunityTrigger.
2. I have created an apex class with the name OpportunityTriggerHandler to calculate business logic for the close date.
3. I have also created a custom label - **Opportunity\_Record\_Link**

to store Opportunity record link.

1. I've explored a few approaches to display it as a hyperlink; however, due to time constraints, I wasn't able to fully implement the solution. As a result, I've displayed the opportunity record link within the error message for now
2. I have also written a test class, **OpportunityTriggerHandlerTest** to cover the unit test cases with assert statements.
3. Please find attached a screen recording, which has demo steps of this solution:

[Question 2 solution video](https://drive.google.com/file/d/1jfGMPEqhD7BeCU22FrebzAXFJJJeoDYw/view)

**Requirement 3**

**Solution:**

1. I have created a GitHub repository-[**SF-Account-Contacts-LWC**](https://github.com/SurbhiB62/SF-Account-Contacts-LWC)
2. I have uploaded all the required components in the repository

**Requirement 4**

**Solution:**

Requirement 4:

Solution:

1. Profiles and Permission Sets: Control object and field-level access.

2. Organization-Wide Defaults (OWD): Set baseline sharing for records.

3. Role Hierarchy: Define access based on user roles.

4. Sharing Rules: Grant additional access if needed.

5. Custom Permissions:Fine-tune access for specific users or profiles.

6. Page Layouts and Record Types: Control UI visibility.

Step-by-Step Best Approach

1. Define Profiles and Permission Sets

- Relationship Managers:

- Create a custom permission (e.g., View Financial Details).

- Assign this permission to a permission set.

- Assign the permission set to Relationship Managers.

- Agents:

- Ensure agents have Read access to the Transaction object but not View All or Modify All.

2. Set Organization-Wide Defaults (OWD)

- Set the Transaction object's OWD to Private.

- This ensures that only the record owner and users above them in the role hierarchy can view the record.

3. Use Role Hierarchy

- Define a role hierarchy that reflects your organization's structure.

- Ensure Relationship Managers are above Agents in the hierarchy to grant them access to all records.

4. Implement Field-Level Security (FLS)

- Restrict access to sensitive fields (e.g., financial details) on the Transaction object.

- Ensure only Relationship Managers can view these fields.

5. Use Sharing Rules (if needed)

- If agents need access to records owned by others in their team, create sharing rules based on criteria (e.g., role, public groups).

6. Assign Records to Agents

- Ensure each Transaction record is assigned to the appropriate agent (via the Owner field).

7. Control UI Visibility with Page Layouts and Record Types

- Create separate page layouts for Relationship Managers and Agents

- Assign the appropriate layout to each profile.

1. Transaction Object

- Contains fields like Owner , Financial Details , and Account.

- Financial Details are sensitive and should only be visible to Relationship Managers.

2. Access Control Logic:

- Relationship Managers have a custom permission to view financial details.

- Agents can only access records they own (via OWD set to Private).

3. Sharing and Visibility:

- Role hierarchy ensures Relationship Managers can view all records.

- Sharing rules can be used to grant additional access to agents if needed.