**JOB RECRUITMENT & HIRING MANAGEMENT SYSTEM**

**NAME: KOLLI JAGAN MOHAN RAO**

**Phase 3: Data Modeling & Relationships**

* **Standard & Custom Objects**:
  + **Standard Objects**: We utilized the standard **User** object by creating lookup relationships to it from our custom objects (Job\_\_c and Interview\_\_c) to assign records to specific users like the Hiring Manager and Interviewer.
  + **Custom Objects**: We built the entire application on a foundation of four new custom objects to store our specialized data: **Job\_\_c**, **Candidate\_\_c**, **Application\_\_c**, and **Interview\_\_c**.
* **Fields**:
  + We created numerous custom fields on our objects to store specific data points. Examples include:
    - **Status\_\_c** on Job\_\_c and Application\_\_c (Picklist)
    - **Email\_\_c** on Candidate\_\_c (Email, Unique)
    - **Resume\_\_c** on Candidate\_\_c (URL)
    - **Hiring\_Manager\_\_c** on Job\_\_c (Lookup to User)
* **Record Types**:
  + We did **not** use Record Types in this project. While they are useful for creating different page layouts and picklist values for different types of records (e.g., "Full-Time Job" vs. "Internship Job"), our project's scope was focused on a single, streamlined process.
* **Page Layouts**:
  + We heavily customized the page layouts for our custom objects. For the **Job\_\_c** record page, we used the **Lightning App Builder** to create a tabbed layout for a better user experience. For the **Candidate\_\_c** record page, we added our custom Lightning Web Component to the sidebar. We also added and removed related lists, such as adding the **"User External Credentials"** list to the User page layout to troubleshoot an issue.
* **Compact Layouts**:
  + While we did not manually edit the compact layouts, Salesforce automatically created them for our custom objects. These layouts control which key fields appear in the highlights panel at the top of a record page.
* **Schema Builder**:
  + We did **not** use the Schema Builder for this project, as we created our objects one by one through the Object Manager. The Schema Builder is a visual tool that would have allowed us to see our entire data model and the relationships between our Job, Candidate, Application, and Interview objects on a single screen.
* **Lookup vs Master-Detail vs Hierarchical Relationships**:
  + **Lookup**: We used a required Lookup relationship to link the **Interview\_\_c** object to the **Application\_\_c** object. We chose a lookup because the Application object was already a detail object in two master-detail relationships and could not be a master to another.
  + **Master-Detail**: We used two Master-Detail relationships to create our junction object, **Application\_\_c**. This created a strong parent-child link to both **Candidate\_\_c** and **Job\_\_c**, ensuring that the application's security is controlled by its parents.
  + **Hierarchical**: We did **not** use a Hierarchical relationship, as this is a special type of lookup used only on the User object to create relationships like "manager-employee".
* **Junction Objects**:
  + The **Application\_\_c** object is the central junction object in our entire data model. It sits between Job\_\_c and Candidate\_\_c and creates a many-to-many relationship between them, allowing one candidate to apply for many jobs, and one job to have many candidates.
* **External Objects**:
  + We did **not** use External Objects. Instead of connecting to an external database with an External Object, we performed a real-time **API callout** in Phase 7 to fetch data from an external system on demand.