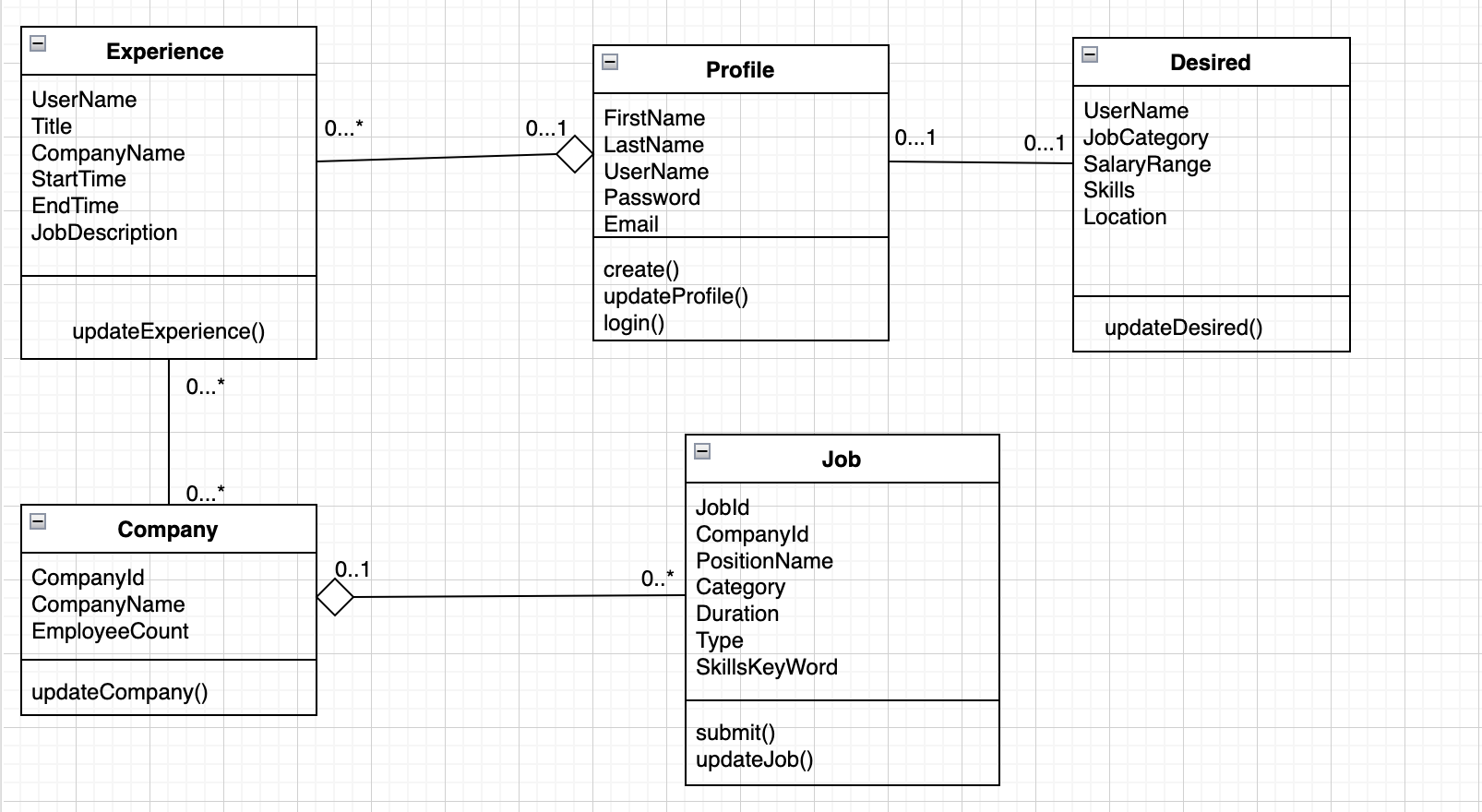
Stage 2

1. Draw your ER/UML diagram and save it as a single markdown or PDF file. Then save this file in the doc folder of your project GitHub repo.



2. Explain your assumptions for each entity and relationship in your model. Discuss why you've modeled something as an entity rather than an attribute of another entity. Describe the cardinality of relationships, like why a student is linked to only one advisor. These assumptions might come from customer requirements or application constraints. Please clarify them.

**Entities and Attributes**

1. **Profile**
   1. **Assumption**: Each user must have a profile to interact with the system, encompassing personal details and authentication information.
   2. **Modeling Decision**: Modeled as a separate entity rather than attributes of Experience or Desired because these details are foundational for the user’s interaction with the system across different modules (job application, profile updates).
2. **Experience**
   1. **Assumption**: Users can have multiple job experiences.
   2. **Modeling Decision**: Experience is an entity since it contains multiple attributes and operations that are distinct to a job role, such as job descriptions and company names.
3. **Company**
   1. **Assumption**: Company information needs to be centralized to ensure consistency across the system when referenced by multiple job experiences.
   2. **Modeling Decision**: Modeled as an entity to maintain independent records of companies that can be referenced by various experiences.
4. **Job**
   1. **Assumption**: Contains information specific to job postings.
   2. **Modeling Decision**: It’s an entity because it encompasses a wide range of attributes and functions that define a job opening.
5. **Desired**
   1. **Assumption**: Users have specific desires or preferences for future job roles.
   2. **Modeling Decision**: Treated as an entity because it includes a variety of preferences that are not directly linked to a user’s past experiences or profile data but are critical for job matching.

**Relationships and Cardinality**

1. **Profile and Experience**
   1. **Cardinality**: One-to-Many (0..1 to 0..\*)
   2. **Assumption**: A user may or may not have job experiences, but each experience relates to exactly one user.
   3. **Rationale**: Reflects that not all users will have job experiences but ensures that any listed experience is clearly linked to a specific user.
2. **Experience and Company**
   1. **Cardinality**: Many-to-One (0..\* to 0..1)
   2. **Assumption**: An experience must be associated with one and only one company, but a company can be associated with many experiences.
   3. **Rationale**: Companies can be linked to multiple experiences reflecting the potential for many roles or positions over time within the same company.
3. **Job and Company**
   1. **Cardinality**: Many-to-One (0..\* to 0..1)
   2. **Assumption**: Each job posting is associated with one company, but a company can have multiple job postings.
   3. **Rationale**: Allows multiple job postings to be associated with the same company, useful for larger organizations with multiple openings.
4. **Profile and Desired**
   1. **Cardinality**: One-to-One (0..1 to 0..1)
   2. **Assumption**: A user can set up a desired job profile that reflects their job preferences.
   3. **Rationale**: Each user can have only one set of job preferences at any given time, simplifying the matching process with potential job openings.

4. Normalize your database. Apply BCNF or 3NF to your schema or show that your schema adheres to one of these normal forms.

For compliance with 3NF, there should be no transitive dependencies. Here we look into our design.

* **Profile**: No transitive dependencies are present. Attributes like *FirstName*, *LastName*, and *Email* do not depend on other non-key attributes.
* **Experience**: This entity has no transitive dependencies. All non-key attributes like *CompanyName* and *JobDescription* depend only on the composite primary key.
* **Company**: The entity does not display any transitive dependency. Attributes depend directly on *CompanyId*.
* **Job**: All attributes depend solely on *JobId*, and no non-key attribute is dependent on another non-key attribute.
* **Desired**: Attributes like *JobCategory*, *SalaryRange*, and Location depend only on *UserName*, indicating the absence of transitive dependencies.

Thus, our design already adheres to 3NF. Therefore, no changes are necessary for normalization to 3NF.

5. Convert your conceptual database design (ER/UML) to the logical design (relational schema). Note that a relational schema is NOT an SQL DDL command.

* Profile(**UserName:** VARCHAR(50) [PK], **FirstName:** VARCHAR(50), **LastName:** VARCHAR(50), **Password:** VARCHAR(100), **Email:** VARCHAR(100))
* Experience(**UserName:** VARCHAR(50) [FK to Profile.UserName], **Title:** VARCHAR(50), **CompanyName:** VARCHAR(100), **StartTime:** DATE, **EndTime:** DATE, **JobDescription:** VARCHAR(1000))
* Company(**CompanyId:** INT [PK], **CompanyName:** VARCHAR(100), **EmployeeCount:** INT)
* Job(**JobId:** INT [PK], **CompanyId:** INT [FK to Company.CompanyId], **PositionName:** VARCHAR(100), **Category:** VARCHAR(50), **Duration:** VARCHAR(50), **Type:** VARCHAR(50), **SkillsKeyWord:** VARCHAR(500))
* Desired(**UserName:** VARCHAR(50) [FK to Profile.UserName], **JobCategory:** VARCHAR(50), **SalaryRange:** VARCHAR(50), **Skills:** VARCHAR(500), **Location:** VARCHAR(100))