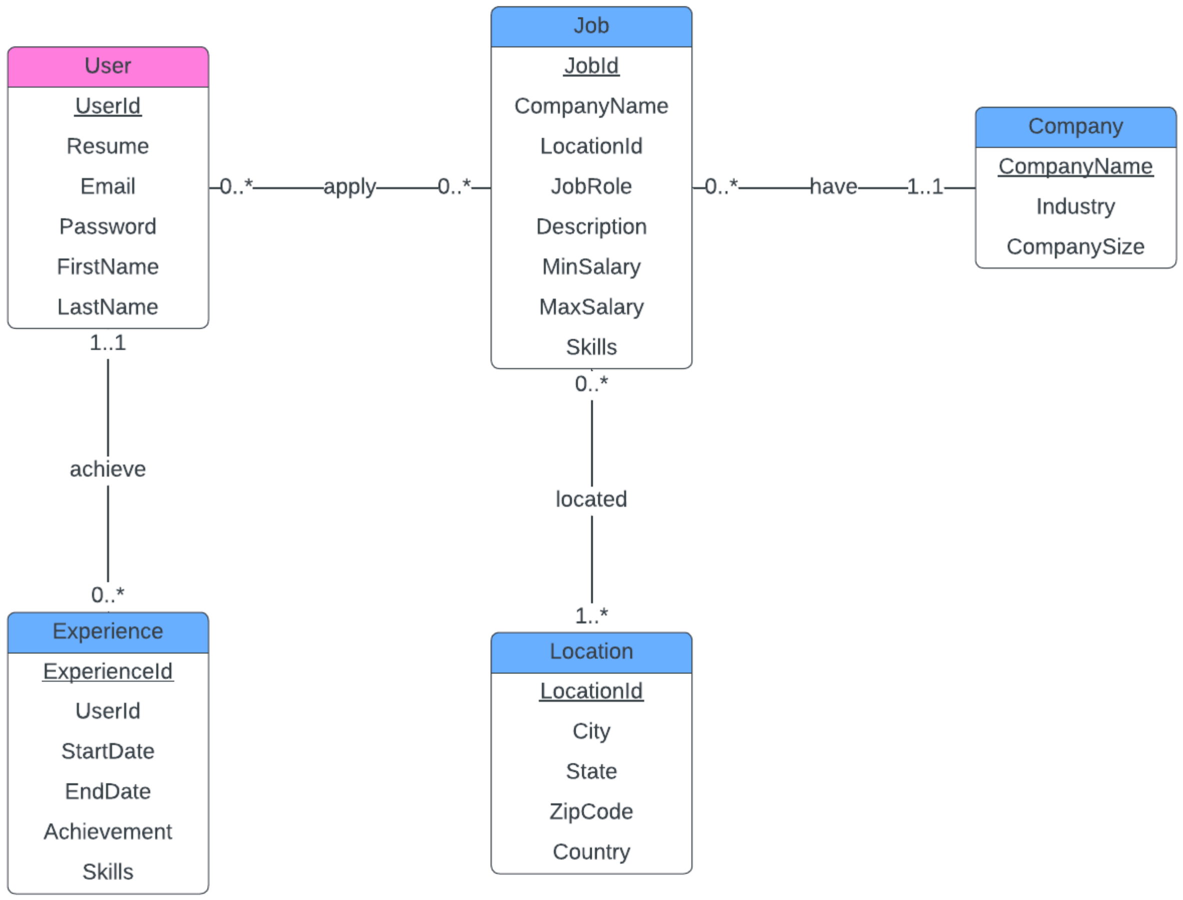
Stage 2: Database Design

UML Diagram



Description of Database Entities and Relationships

User

A user represents a person using the platform. Each user has unique information about them to access their account. All the attributes in this table are core characteristics of a user in our system,

UserId: This is a **primary key** to store a unique userId, stored as an **INT**.
Resume: This stores a text representation of a resume as a **VARCHAR(MAX)**.
Email: This stores an email address used to login as a **VARCHAR(255).**
Password: This stores a hashed password to login as a **VARCHAR(255).**
FirstName: This stores a user's first name as a **VARCHAR(255).**
LastName: This stores a user's last name as a **VARCHAR(255).**

Relationships:

- Users will have one-to-many relationships with Experience as one user can have multiple past experiences
- Users will have a many-to-many relationship with Job, as many users can have multiple job applications.

Job

A job represents a job listing on the platform. It has attributes such as a job role, description, salary.

JobId: This is a **primary key** to store a unique JobId, stored as an **INT**.
CompanyName: This is a foreign key linking the job to the Company entity, representing the company offering the job. Stored as an **VARCHAR(255)**.
JobRole: This represents the title or role of the job (e.g., Software Engineer, Product Manager). It is stored as a **VARCHAR(255)**.
LocationId: This is a foreign key linking the job to a **Location**, indicating where the job is based. Stored as an **INT**.
Description: This field contains a detailed description of the job responsibilities, expectations, and requirements. It is stored as a **VARCHAR(255)**.
MinSalary: This represents the minimum salary for the position, stored as an **INT**.
MaxSalary: This represents the maximum salary for the position, stored as an **INT**.
Skills: This contains a list of skills required for the job, stored as a **VARCHAR(MAX)**, assuming a text-based representation of skills.

Relationships:

- Job will have a many-to-one relationship with *Company,* connected by FK CompanyName, as a job can only belong to one company.
- Job will have a many-to-many relationship with *Location*, connected by FK LocationId, as each job can have multiple locations (multiple possible offices for example).
- Job will have a many-to-many relationship with *Users* because many users can apply to the same job.

Company

A company represents a company that is posting a job. A company is treated as an entity since it holds key information about the business that is posting the job.

CompanyName: This represents the name of the company. Stored as a **VARCHAR(255)**.
Industry: This indicates the sector or industry in which the company operates (e.g., Technology, Healthcare). Stored as a **VARCHAR(255)**.
CompanySize: This represents the size of the company (e.g., number of employees). Stored as an **INT**.

Relationships:

- A company will have a one-to-many relationship with *Job*, as a company can post multiple jobs, but each job is associated with only one company.

Location

A location represents a physical or remote address where jobs are based. It stores details about the city, state, and country.

LocationId: This is the primary key to uniquely identify a location. Stored as an **INT**.
City: This represents the city where the job or company is located. Stored as a **VARCHAR(255)**.
State: This represents the state or region where the job or company is located. Stored as a **VARCHAR(255)**.
ZipCode: This stores the postal code for the location. Stored as a **VARCHAR(10)**.
Country: This represents the country of the location. Stored as a **VARCHAR(255)**.

Relationships:

- A location will have a many-to-many relationship with *Job*, as a location can host multiple jobs, but each job can be at multiple locations.

Experience

Experience represents a user's past work history, including the skills and achievements gained from that job.

ExperienceId: This is the primary key to uniquely identify an experience. Stored as an **INT**.
UserId: This is a foreign key linking the experience to a specific user. Stored as an **INT**.
StartDate: This represents the start date of the experience. Stored as a **DATE**.
EndDate: This represents the end date of the experience. Stored as a **DATE**.

Achievement: This represents any notable achievements the user gained during their experience. Stored as a **VARCHAR(MAX)**.
Skills: This contains a list of skills the user gained or used in this experience. Stored as a **VARCHAR(MAX)**.

Relationships:
- Experience will have many-to-one relationships with Users as one user can have multiple past experiences

UserJob (Junction Table)

The **UserJob** table manages the many-to-many relationship between **User** and **Job**, representing job applications.

Attributes:
UserId: This is a foreign key linking to the User entity. Stored as an INT.
JobId: This is a foreign key linking to the Job entity. Stored as an INT.
ApplicationDate: The date when the user applied for the job. Stored as a DATE.
Status: The current status of the job application (e.g., Applied, Interviewing, Hired). Stored as a VARCHAR(50).

Relationships:

Each application will have one user so Many-to-One with User.

Each application will have one job so Many-to-One with Job.

JobLocation (Junction Table)

The JobLocation table manages the many-to-many relationship between Job and Location.

Attributes:
JobId: This is a foreign key linking to the Job entity. Stored as an INT.
LocationId: This is a foreign key linking to the Location entity. Stored as an INT.

Relationships:
Many-to-One with Job: Each record associates one job with a location.
Many-to-One with Location: Each record associates one location with a job.

Normalization - BCNF

User

- Functional dependencies
  UserId → (Resume, Email, Password, FirstName, LastName)
  Email → (UserId, Resume, Password, FirstName, LastName)
- Superkey : UserId, Email
- BCNF because the superkeys determine all other attributes

Job

- Functional dependencies
  JobId → (CompanyName, JobRole, Description, MinSalary, MaxSalary, Skills)
- Superkey : JobId
- It violates BCNF because the superkey determines all other attributes but LocationId
  ➩ We make a new table, JobLocation

Location

- Functional dependencies
  LocationId → (City, State, ZipCode, Country)
  (City, State, ZipCode) → LocationId
- Superkey : LocationId, (City, State, ZipCode)
- BCNF because the superkeys determine all other attributes

Company

- Functional dependencies
  CompanyName → (Industry, CompanySize)
- Superkey : CompanyName
- BCNF because the superkey determines all other attributes

Experience

- Functional dependencies
  ExperienceId → (UserId, StartDate, EndDate, Achievement, Skills)

UserId → Skills
- Superkey : ExperienceId
- BCNF because the superkey determines all other attributes and Skills is an attribute in the User table where UserId is the primary key

UserJob

- A new table for the relationship between User and Job entities
- Functional Dependencies:
  (UserId, JobId) → (ApplicationDate, Status)
- Superkey: (UserId, JobId)
- BCNF: The table is in BCNF because the composite primary key (UserId, JobId) determines all other attributes.

JobLocation

- A new table to resolve troublesome functional dependency from Job
- Functional Dependencies:
  (JobId, LocationId) → (no other attributes)
- Superkey: (JobId, LocationId)
- BCNF: The table is in BCNF because the composite primary key (JobId, LocationId) is the only determinant.

Relational Schema

- User

User(UserId: INT [PK], Resume: VARCHAR(MAX), Email: VARCHAR(255), Password: VARCHAR(255), FirstName: VARCHAR(255), LastName: VARCHAR(255))

- Experience

Experience(ExperienceId: INT [PK], UserId: INT [FK to User.UserId], StartDate: DATE, EndDate: DATE, Achievement: VARCHAR(MAX), Skills: VARCHAR(MAX))

- ## Company

  Company(CompanyName: VARCHAR(255) [PK], Industry: VARCHAR(255), CompanySize: INT)

- ## Job

  Job(JobId: INT [PK], CompanyName: VARCHAR(255) [FK to Company.CompanyName], JobRole: VARCHAR(255), Description: VARCHAR(255), MinSalary: INT, MaxSalary: INT, Skills: VARCHAR(MAX))

- ## Location

  Location(LocationId: INT [PK], City: VARCHAR(255), State: VARCHAR(255), ZipCode: VARCHAR(10), Country: VARCHAR(255))

- ## UserJob

  UserJob(UserId: INT [PK, FK to User.UserId], JobId: INT [PK, FK to Job.JobId], ApplicationDate: DATE, Status: VARCHAR(50))

- ## JobLocation

  -JobLocation(JobId: INT [PK, FK to Job.JobId], LocationId: INT [PK, FK to Location.LocationId])