**My prompt--------**Design a database for the application using the ER diagram (ERD) model. Include all the entities, attributes, and relationships needed for the application. The database design should meet the requirements of ALL of the features of your application. That is, you should see how your ERD supports the data needed for each of the features in the System Requirements document for your application. 2. Design a data dictionary that will list and describe the fields for the entities in the diagram. See the tutorial below for an overview of data dictionaries. Data Dictionary Tutorials PointLinks to an external site. 3. Use an AI tool (ChatGPT) to help you design your ERD. This is a requirement. However, it is your responsibility to submit a correct ERD. Work with the tool to develop something that is correct. If it is not correct, it is not the tool's fault, it is yours. Submit the prompts (text, pictures) you use to get to your final ERD, or the closest ERD the tool got until you had to manually complete it. For this deliverable, you must provide the prompts you used. You will demonstrate that during your presentation. See the examples in this module, ChatGPT4o and ChatGPT and ERDs. If the tool just generates a keyboard character diagram, use that to draw a diagram. Diagrams can be drawn with https://app.diagrams.net/Links to an external site. or something similar. See GotItERD.pdf in this Module for an example ER Diagram. 4. Prepare a presentation discussing the E-R diagram and your use of AI (ChatGPT). Only 1 team member, the Scrum Master, needs to submit the design. Keep in mind, this is just the Database Design. You do not need to write any code or build anything. We want to make sure the design is correct before we start building. Choose a Scrum master for this Sprint Cycle. The Scrum master will arrange meetings, track progress, record decisions, and communicate with the customer (Dr. Allen). Create a suitable report that will document your meetings. You should have at least 4 meetings that include all team members. The meetings should be spread out during the entire sprint cycle (not all grouped at the end). You will upload the meeting document to the Canvas site no later than the day after the meeting. The Scrum master will also give the presentation discussing the design and implementation. the document is the application plans and the file is the requirements  
  
**ChatGPT response-----------** Steps to Complete the ER Diagram and Data Dictionary for Career Compass

Step 1: Identify the Key Entities from the Requirements

Based on the provided system requirements for Career Compass, we can derive the following **main entities**:

**Career Compass ERD**

**Users**

**user\_id** PK  
email  
password  
role (Student, Recruiter, Admin)

**Students**

**student\_id** PK, FK (user\_id)  
first\_name  
last\_name  
skills  
resume (optional)

**Recruiters**

**recruiter\_id** PK, FK (user\_id)  
company\_name

**Jobs/Positions**

**position\_id** PK  
**recruiter\_id** FK (recruiters)  
title  
description  
requirements  
location  
application\_deadline

**Applications**

**application\_id** PK  
**student\_id** FK (students)  
**position\_id** FK (jobs)  
application\_date  
status (Submitted, Under Review, Interview, Offered, Rejected)

**Notifications**

**notification\_id** PK  
**user\_id** FK (users)  
message  
timestamp  
status (Read/Unread)

**Admin Moderation**

**admin\_id** PK, FK (users)  
action\_taken  
timestamp

**Step 2: Design the ER Diagram (ERD)**

I will generate an ER diagram that visually represents these entities and their relationships.

**Step 3: Create a Data Dictionary**

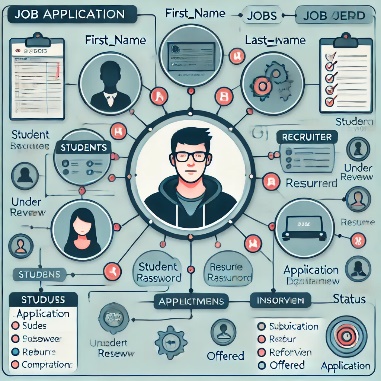
The **data dictionary** will list all tables, attributes, data types, and descriptions of the fields.

**Step 4: AI Tool Usage and Prompting**

I will prompt an AI tool (ChatGPT) to generate an initial ER diagram and refine it iteratively based on validation.

**Next Steps**

1. I will generate the **ER Diagram**.
2. I will create the **Data Dictionary**.
3. I will iterate on the **AI-generated ERD and test it for correctness**.
4. I will prepare a **report documenting the prompts used for AI-generated ERD**.



Here is the generated **Entity-Relationship Diagram (ERD)** for the *Career Compass* job application platform. Let me know if you need any modifications or if you’d like me to refine any relationships!

**My Response-----** can you make it in this type of format? **GotItERD-1.pdf**

**ChatGPT response-----------** Yes! I can format the Career Compass ERD in the same structured tabular format as shown in *GotItERD-1.pdf*. I will now create a structured ERD with primary keys (PK), foreign keys (FK), and attributes formatted similarly.

Let me generate this for you.

**Career Compass ERD**

**Users**

**user\_id PK  
email  
password  
role (Student, Recruiter, Admin)**

**Students**

**student\_id PK, FK (user\_id)  
first\_name  
last\_name  
skills  
resume (optional)**

**Recruiters**

**recruiter\_id PK, FK (user\_id)  
company\_name**

**Jobs/Positions**

**position\_id PK  
recruiter\_id FK (recruiters)  
title  
description  
requirements  
location  
application\_deadline**

**Applications**

**application\_id PK  
student\_id FK (students)  
position\_id FK (jobs)  
application\_date  
status (Submitted, Under Review, Interview, Offered, Rejected)**

**Notifications**

**notification\_id PK  
user\_id FK (users)  
message  
timestamp  
status (Read/Unread)**

**Admin Moderation**

**admin\_id PK, FK (users)  
action\_taken  
timestamp**

**Relationships**

* **Users have a one-to-many relationship with Students and Recruiters.**
* **Recruiters can post multiple Jobs.**
* **Students can apply to multiple Jobs (many-to-many relationship through Applications).**
* **Users receive Notifications.**
* **Admins moderate Job Postings.**

**Relationships**

**Users have a one-to-many relationship with Students and Recruiters.**

**Recruiters can post multiple Jobs.**

**Students can apply to multiple Jobs (many-to-many relationship through Applications).**

**Users receive Notifications.**

**Admins moderate Job Postings.  
  
can you give a bit more clarity?**

**I have formatted the Career Compass ERD in the same structured tabular style as *GotItERD-1.pdf*. Let me know if you need any modifications or additional details!**

**My response----------** can you kind of draw it out a bit so i can see whats connected to what and where  
**ChatGPT response-----------** A blue and white diagram with text and images

Description automatically generated

Here is the **visual ER diagram** for the *Career Compass* job application platform. It clearly shows the connections between entities, including primary keys and foreign key relationships. Let me know if you need any refinements!

**My Response-----------** Relationships Users have a one-to-many relationship with Students and Recruiters. Recruiters can post multiple Jobs. Students can apply to multiple Jobs (many-to-many relationship through Applications). Users receive Notifications. Admins moderate Job Postings. can you give a bit more clarity?

**ChatGPT response---------**Certainly! Below is a more **detailed explanation** of the **relationships** between entities in the *Career Compass* ERD.

**1. Users → Students & Recruiters (One-to-Many)**

* The **Users** table contains **all** users, including **students, recruiters, and admins**.
* Each user has a **role** (Student, Recruiter, or Admin).
* **A single user** can either be a **Student** or a **Recruiter**, but not both.
* **One User → One Student** *(if the user is a student)*
* **One User → One Recruiter** *(if the user is a recruiter)*

👉 **Example:**

* A user with user\_id = 1001 might be a student (student\_id = 2001).
* Another user with user\_id = 1002 might be a recruiter (recruiter\_id = 3001).

**2. Recruiters → Jobs (One-to-Many)**

* A **recruiter** can post **multiple job positions**.
* Each job posting is linked to **one recruiter**.
* This means **a single recruiter can manage multiple job listings**, but each job belongs to only **one recruiter**.

👉 **Example:**

* A recruiter with recruiter\_id = 3001 can post **three job positions**:
  + position\_id = 4001 (Software Engineer)
  + position\_id = 4002 (Data Analyst)
  + position\_id = 4003 (Marketing Intern)

**3. Students → Jobs (Many-to-Many via Applications)**

* **Students** can apply to **multiple jobs**.
* **Jobs** can have **multiple student applicants**.
* Since a direct **many-to-many** relationship is not possible in a relational database, we use an **intermediary table** called **Applications**.
* The **Applications** table has:
  + student\_id (FK → Students)
  + position\_id (FK → Jobs)
  + status (Submitted, Under Review, Interview, Offered, Rejected)

👉 **Example:**

* A student with student\_id = 2001 applies for two jobs:
  + position\_id = 4001 (Software Engineer)
  + position\_id = 4002 (Data Analyst)
* The **Applications table** will store:
  + (application\_id = 5001, student\_id = 2001, position\_id = 4001, status = Submitted)
  + (application\_id = 5002, student\_id = 2001, position\_id = 4002, status = Interview)

**4. Users → Notifications (One-to-Many)**

* Each **user** can receive **multiple notifications**.
* Notifications are linked to **users** by user\_id.
* Notifications might include:
  + Job application updates
  + New job postings that match a student’s profile
  + Recruiter messages
  + System alerts

👉 **Example:**

* A student with user\_id = 1001 receives:
  + notification\_id = 6001 (Your application for Software Engineer is under review.)
  + notification\_id = 6002 (New job alert: Data Scientist position available.)

**5. Admins → Job Postings (One-to-Many)**

* **Admins** moderate job postings to ensure they meet quality guidelines.
* Admins can **approve, reject, or flag** job postings.
* Each moderation action is recorded in an **Admin Moderation** table.

👉 **Example:**

* An admin with admin\_id = 7001 reviews:
  + position\_id = 4003 (Marketing Intern) → Approved
  + position\_id = 4004 (Unverified Job) → Rejected

**Final Clarification of Relationships**

| **Relationship** | **Type** | **Explanation** |
| --- | --- | --- |
| **Users → Students** | One-to-One | A user can be a student. |
| **Users → Recruiters** | One-to-One | A user can be a recruiter. |
| **Recruiters → Jobs** | One-to-Many | A recruiter can post multiple jobs. |
| **Students → Jobs** | Many-to-Many | Students apply to multiple jobs, and jobs have multiple applicants (handled via **Applications**). |
| **Users → Notifications** | One-to-Many | Users receive multiple notifications. |
| **Admins → Job Postings** | One-to-Many | Admins moderate multiple job postings. |

**Would You Like a Visual Update?**

I can refine the **ERD diagram** to reflect these relationships more clearly. Let me know if you’d like any further adjustments! 🚀

**My response-----------** would the relationship from applications to jobs be one to one?

**ChatGPT response---------** No, the relationship from Applications to Jobs is many-to-one, not one-to-one. Here's why:

Correct Relationship: Applications → Jobs (Many-to-One)

* A single job (position) can receive many applications from different students.
* Each application is linked to one specific job.

Breaking it Down:

* One job (position\_id) can have many applications (application\_id).
* One application (application\_id) corresponds to one job (position\_id).

Example:

Imagine a job posting for Software Engineer (position\_id = 4001).

* Student A (student\_id = 2001) applies → Creates application\_id = 5001.
* Student B (student\_id = 2002) applies → Creates application\_id = 5002.
* Student C (student\_id = 2003) applies → Creates application\_id = 5003.

Each application is unique but points to the same job.

Thus, the relationship Applications → Jobs is Many-to-One *(many applications per job, one job per application).*

Would It Ever Be One-to-One?

A one-to-one relationship would mean:

* Each job could only ever have one application, which isn't realistic for a job board.
* If this were the case, a job would "disappear" after one person applies.

Since jobs typically receive multiple applications, a many-to-one relationship is correct.

Let me know if you need further clarification! 🚀

**My response----------** can you clarify all the relationships on more time?

**ChatGPT response**---------- Certainly! Below is a **detailed breakdown of all relationships** in the **Career Compass ERD**, clarifying how each entity connects.

**Final Relationship Clarifications**

**1. Users → Students (One-to-One)**

* Each **User** can be a **Student**, but not all users are students.
* A **Student** must have exactly **one** User account.

✅ **One User → One Student**

🔹 **Example:**

* user\_id = 1001 is linked to student\_id = 2001 (Same person).
* A **User** who is a **Recruiter** does not have a student\_id.

**2. Users → Recruiters (One-to-One)**

* Each **User** can be a **Recruiter**, but not all users are recruiters.
* A **Recruiter** must have exactly **one** User account.

✅ **One User → One Recruiter**

🔹 **Example:**

* user\_id = 1002 is linked to recruiter\_id = 3001 (Same person).
* A **User** who is a **Student** does not have a recruiter\_id.

**3. Recruiters → Jobs (One-to-Many)**

* A **Recruiter** can post **multiple Jobs**.
* Each **Job** belongs to **only one Recruiter**.

✅ **One Recruiter → Many Jobs**

🔹 **Example:**

* recruiter\_id = 3001 posts:
  + position\_id = 4001 (Software Engineer)
  + position\_id = 4002 (Data Analyst).
* Each job is **owned by only one recruiter**.

**4. Students → Jobs (Many-to-Many via Applications)**

* A **Student** can apply to **multiple Jobs**.
* A **Job** can have **multiple Student applicants**.
* **Since direct many-to-many relationships are not possible in relational databases, we use an intermediary table: Applications.**

✅ **Many Students → Many Jobs (via Applications Table)**

🔹 **Example:**

* **Student A (student\_id = 2001) applies to:**
  + position\_id = 4001 (Software Engineer)
  + position\_id = 4002 (Data Analyst).
* **Job (position\_id = 4001) receives applications from:**
  + student\_id = 2001 (Alice)
  + student\_id = 2002 (Bob)
  + student\_id = 2003 (Charlie).

Since **one job can receive multiple applications** and **one student can apply to multiple jobs**, this is a **many-to-many** relationship.

**5. Applications → Jobs (Many-to-One)**

* Each **Application** is linked to **one specific Job**.
* Each **Job** can have **many Applications**.

✅ **Many Applications → One Job**

🔹 **Example:**

* application\_id = 5001 applies for position\_id = 4001 (Software Engineer).
* application\_id = 5002 applies for the same job (position\_id = 4001).

Each application is **unique but points to a single job**.

**6. Applications → Students (Many-to-One)**

* Each **Application** is linked to **one specific Student**.
* Each **Student** can have **multiple Applications**.

✅ **Many Applications → One Student**

🔹 **Example:**

* student\_id = 2001 (Alice) submits:
  + application\_id = 5001 (Software Engineer)
  + application\_id = 5002 (Data Analyst).

Each application belongs to **one** student, but students can submit **multiple applications**.

**7. Users → Notifications (One-to-Many)**

* Each **User** can receive **multiple Notifications**.
* Each **Notification** is linked to **one specific User**.

✅ **One User → Many Notifications**

🔹 **Example:**

* user\_id = 1001 receives:
  + notification\_id = 6001 (Your application is under review)
  + notification\_id = 6002 (New job alert: Data Scientist available).

Each notification is **linked to one user**.

**8. Admins → Jobs (One-to-Many)**

* **Admins moderate multiple job postings.**
* Each **moderation action** is linked to **one Admin**.

✅ **One Admin → Many Job Moderation Actions**

🔹 **Example:**

* admin\_id = 7001 moderates:
  + position\_id = 4003 (Marketing Intern) → Approved
  + position\_id = 4004 (Unverified Job) → Rejected.

Each **moderation action** belongs to **one Admin**.

**Final Relationship Summary**

| **Relationship** | **Type** | **Explanation** |
| --- | --- | --- |
| **Users → Students** | One-to-One | A user can be a student. |
| **Users → Recruiters** | One-to-One | A user can be a recruiter. |
| **Recruiters → Jobs** | One-to-Many | A recruiter can post multiple jobs. |
| **Students → Jobs** | Many-to-Many | Students apply to multiple jobs, and jobs have multiple applicants (**via Applications Table**). |
| **Applications → Jobs** | Many-to-One | Many applications belong to one job. |
| **Applications → Students** | Many-to-One | Many applications belong to one student. |
| **Users → Notifications** | One-to-Many | Users receive multiple notifications. |
| **Admins → Jobs** | One-to-Many | Admins moderate multiple job postings. |

**Would You Like a Final ERD Update?**

I can refine the **ERD diagram** to reflect all these relationships **visually**. Let me know if you’d like any **further adjustments or confirmations!** 🚀

**My response---------** can you Design a data dictionary that will list and describe the fields for the entities in the diagram.

**ChatGPT response------------- Career Compass ERD**

**Users**

**user\_id PK  
email  
password  
role (Student, Recruiter, Admin, Career Counselor)**

**Students**

**student\_id PK, FK (user\_id)  
first\_name  
last\_name  
skills  
resume (optional)**

**Recruiters**

**recruiter\_id PK, FK (user\_id)  
company\_name**

**Career Counselors**

**counselor\_id PK, FK (user\_id)  
specialization**

**Jobs/Positions**

**position\_id PK  
recruiter\_id FK (recruiters)  
title  
description  
requirements  
location  
application\_deadline**

**Applications**

**application\_id PK  
student\_id FK (students)  
position\_id FK (jobs)  
application\_date  
status (Submitted, Under Review, Interview, Offered, Rejected)**

**Notifications**

**notification\_id PK  
user\_id FK (users)  
message  
timestamp  
status (Read/Unread)**

**Admin Moderation**

**admin\_id PK, FK (users)  
action\_taken  
timestamp**

**Counselor Sessions**

**session\_id PK  
counselor\_id FK (career\_counselors)  
student\_id FK (students)  
session\_date  
notes**

**Feedback**

**feedback\_id PK  
application\_id FK (applications)  
recruiter\_id FK (recruiters)  
comments  
rating**

**Relationships**

* **Users have a one-to-one relationship with Students, Recruiters, and Career Counselors (each user has only one role).**
* **Recruiters have a one-to-many relationship with Jobs (a recruiter can post multiple job listings, but each job is tied to one recruiter).**
* **Students have a many-to-many relationship with Jobs (handled via the Applications table, as students can apply to multiple jobs and each job can have multiple applicants).**
* **Applications have a many-to-one relationship with Jobs (each application is tied to one specific job, but a job can have many applications).**
* **Applications have a many-to-one relationship with Students (each application belongs to one student, but students can submit multiple applications).**
* **Users have a one-to-many relationship with Notifications (each user can receive multiple notifications).**
* **Admins have a one-to-many relationship with Job Postings (admins can moderate multiple job postings).**
* **Career Counselors have a one-to-many relationship with Counselor Sessions (counselors can schedule multiple sessions with different students).**
* **Career Counselors have a one-to-many relationship with Feedback (counselors can provide feedback on multiple applications).**
* **Recruiters have a one-to-many relationship with Feedback (recruiters can provide feedback on multiple student applications).**