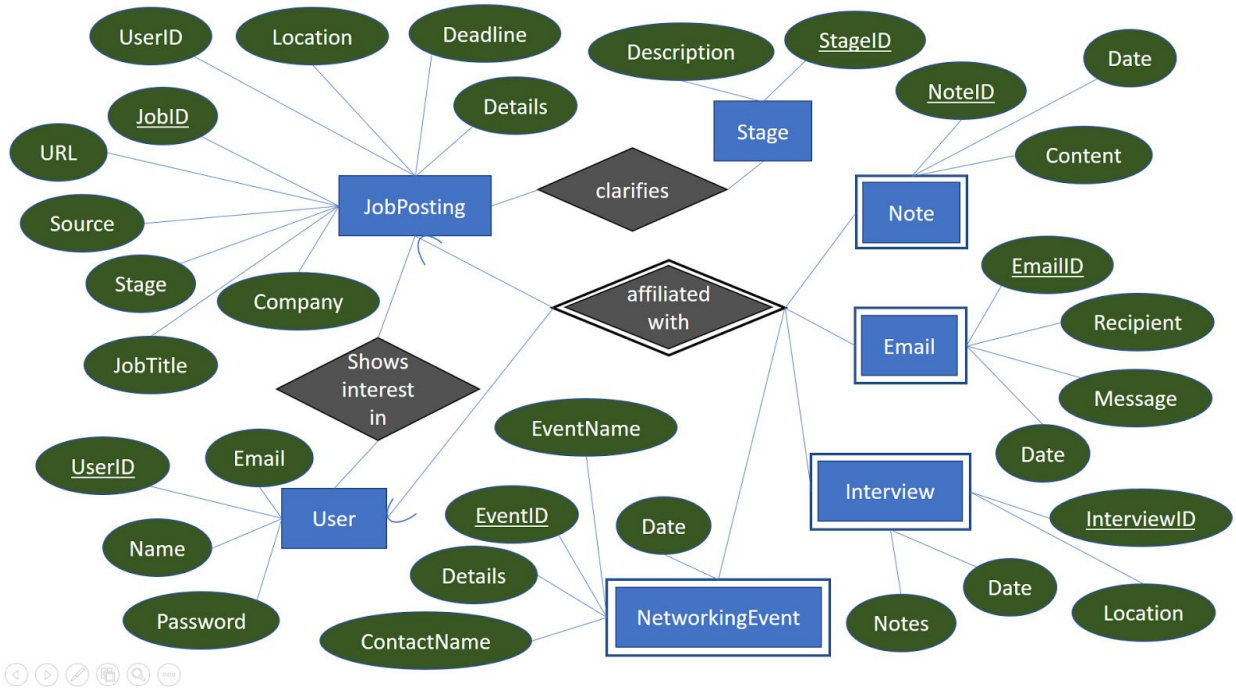
Databases Final Project
Carolyn Bergdolt, Jamie Maher, Troy Prince, Erin Turley

1. Our project Catchit allows a user to easily keep track of ongoing job applications. A user can create an account and view their dashboard, a feed that lists current applications. Users can manually add applications, or they can add an application through the Google Chrome plugin for Indeed and Glassdoor. For each job, a user can easily keep track of notes, emails, events, or interviews associated with the job. Overall, Catchit is an organized means of keeping track of the otherwise disorganized job search process.

2. As our group members spent the semester going through the job search process, we found it difficult at times to keep track of positions we had applied to through various job boards, or specific communications we had had with various companies. A site devoted to an organized means of applying for a job would have been useful to have while going through the search process.

3a. Our ER diagram is displayed below. We had seven different relations, including several weak entity sets. Each relation has an auto-incrementing ID as an attribute that serves as the primary key.




Schema:

```
JobPosting(JobID, UserID, JobTitle, Company, Deadline, URL, Source, Details,
Location, Stage)
User(UserID, Email, Name, Password)
Stage(StageID, Description)
Note(JobID, UserID, NoteID, Content, Date)
```

```
Email(JobID, UserID, EmailID, Recipient, Message, Date)
Interview(JobID, UserID, InterviewID, Date, Location, Notes)
NetworkingEvent(JobID, UserID, EventID, EventName, ContactName, Details,
Date)
```

3b. We obtained the data through two primary means: a user could manually input information for a position (company, location, title, deadline, source, details, and URL link), or the user could use the Chrome plugin, which sends an application's information from Indeed or Glassdoor to the Catchit database for the given user.

4. The functionalities of the project include keeping track of all the job entries in the database, but also organizing notes and correspondences associated with each job. Users are able to manually add jobs to the database, edit jobs, view jobs, add notes or correspondences, edit notes or correspondences, delete notes or correspondences, delete jobs, create a new account, receive email notifications for upcoming job application deadlines, and add jobs on Glassdoor and Indeed with the CatchIt plugin.

5a. One basic function is displaying the jobs for a particular user on their dashboard. The exact SQL query is

```
select g.JobID, g.Company, g.JobTitle, g.Location, g.Deadline, g.Details,
s.Description
from JobPosting g, Stage s
where 1=1
and g.Stage=s.StageID
and g.UserID=?
```

5b. The steps of the data flow are as follows. First, the UserID is obtained from the user logging in to the website. Once they have logged in, their UserID is saved in the session data. This is what allows the user's experience to remain persistent. After the user logs in, they are automatically taken to the dashboard. When in the dashboard, the prior query runs, with the g.UserID being set equal to the session's user ID. This query returns all the information about any jobs that the logged in user is interested in. It is displayed on the webpage by populating an html table with the information returned from the query.

6. One of the advanced functions is the cron job that runs a python script to find upcoming job application deadlines in the database, and alert the job owner to that deadline by emailing them. This is considered advanced because the python script must properly connect both to the database, and to the CatchIt email account we created. The rest of the python script finds any jobs that have a deadline equal to the current date (the script runs at midnight) and saves all the resulting jobs to an array. The script then goes through the array, emailing the owner of each of the various jobs.
Another advanced function is the CatchIt Chrome plugin. This allows the user to automatically add jobs they find on Indeed or Glassdoor to their list of jobs. This is considered advanced

because it requires connecting the Chrome plugin to the database, and parsing the individual pages for the necessary information. Glassdoor and Indeed both had different formats for their website, so different methods of parsing were required for each.

7. The biggest technical challenge we encountered was getting the data from the Chrome plugin into the database. The problem was that the plugin did not have access to the database because it was not running on the dsg1 server. Additionally, giving the plugin access would not have been secure or a good paradigm. Therefore, we had to create an API on the dsg1 server that the plugin would send POST request too and would subsequently put the data in the database. Because of the amount of data being sent (particularly in the description of the job), the data needed to be sent in a POST and not as part of the URL in a GET. Constructing the API on dsg1 simply required the values to be extracted from the POST and to be uploaded to the server. Properly sending the data in the POST was probably the more difficult part.

8. Everything went as we planned. We were successful in creating our advanced functions, which is what we were most concerned about. Some potential improvements we could have had were to make the plugin be able to work on LinkedIn pages or GoIrish pages. It would have also have been useful to have some kind of sort function on the dashboard. However, we met all the requirements that we originally set out to accomplish with the design specifications.

9. As far as the final division of labor goes, we were equally involved in the initial setup of the Database and beginning stages of the project. As we made more progress, we were able to work individually on certain elements, while still helping with other aspects. In general, the areas where people spent the most time working on are:

Troy - Creating the Chrome plugin and the API on dsg1
Jamie - bootstrap and design, python email script, help with plugin
Carolyn - job insertion/updates, correspondence insertion/updates, crontab settings
Erin - dashboard dropdowns, displaying correspondence information