# *Job Schema-ers*

## Schema-ers

332 Databases & File Structures

Due: 5/16/2022

https://github.com/TelloVisionGames/Relational-Database-Design

Corey New
Gabe Warkentin
Jason Zhu
Josh Lollis
Kendrick Ngo
Shivam Sudame

# Index

# Introduction

In this project, our task was to create a Job Posting application website that adheres to a list of requirements in order to reach a final product. That product is where users are able to navigate through an interface and store important information regarding a specific job. The main objective was for any user to manipulate their own information from a secure database with ease. In order to do that, we were able to follow a specific database design process so that each requirement is met for an application design. The following will display the process for making this application possible.

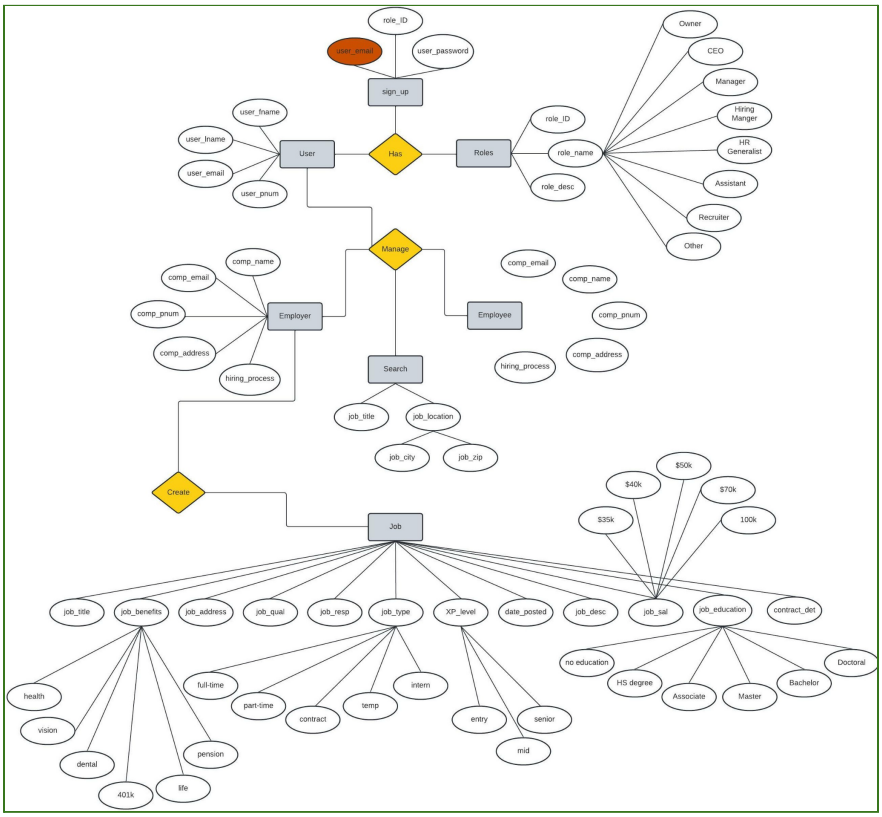
# Database Design Process

## Requirements

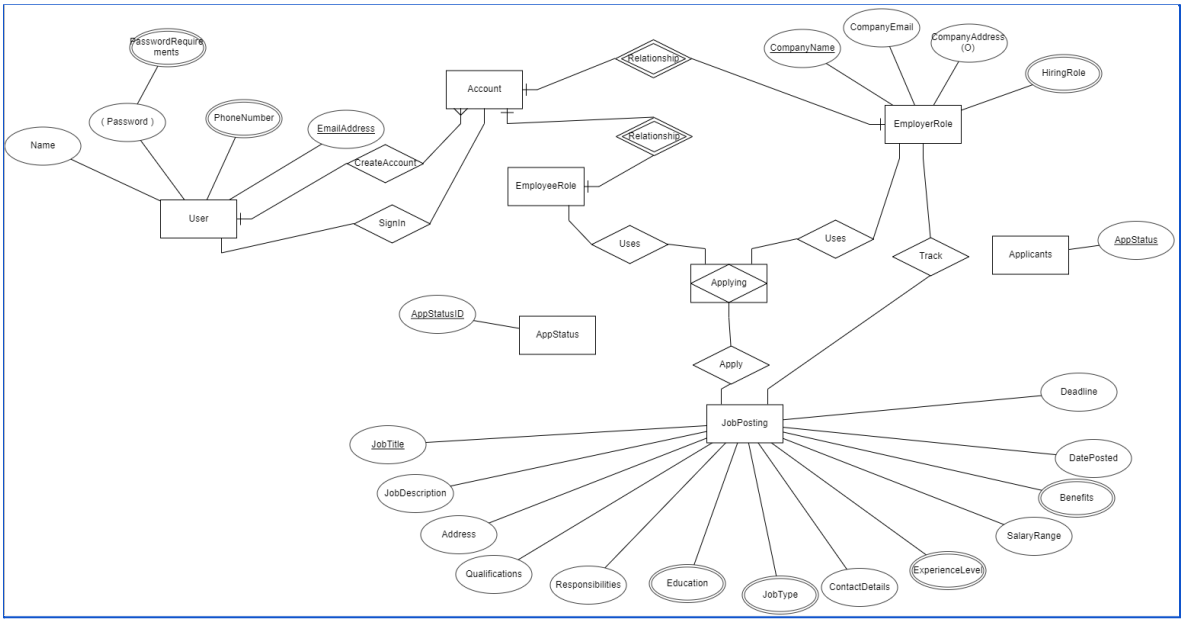
- Build an ER Diagram
    - All team members created their own ER diagram to come to an agreement in order to complete the final relational model
- Build a Relational Model
    - Once all diagrams were agreed upon, requirements were re-evaluated to fit the main business model
- Normalize the database to 3NF
    - This normalization made our database easier to access certain data within our application
- Create physical database model
    - Fill database with sample data to test the demo application

# ER Diagrams

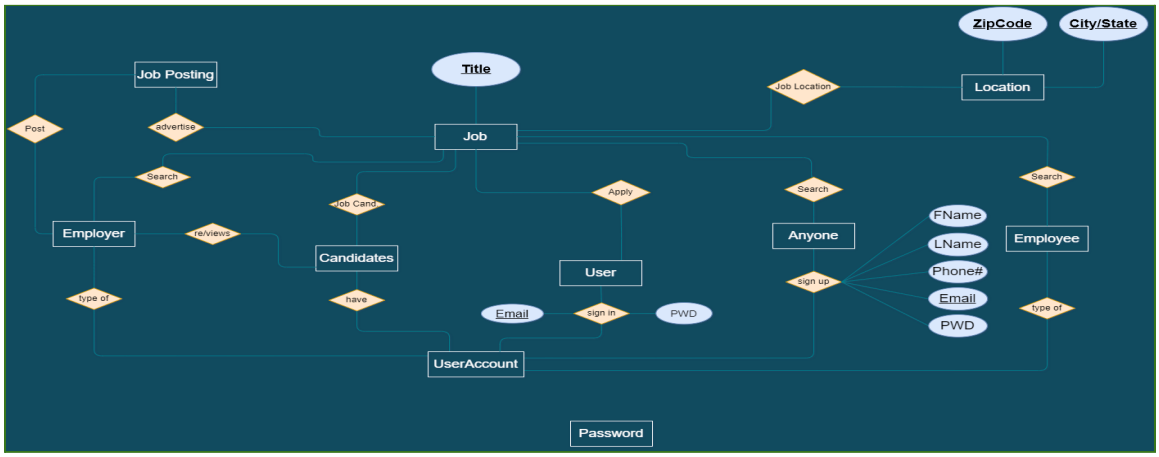
## Kendrick Ngo



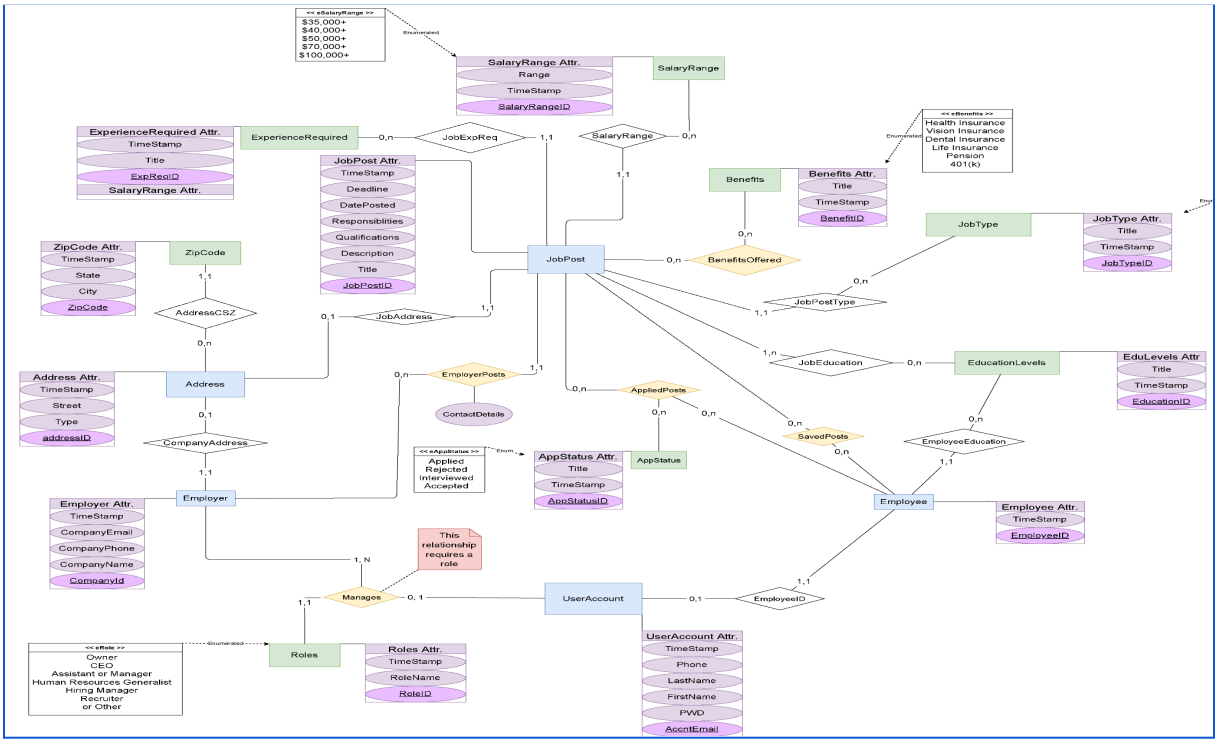

## Jason Zhu



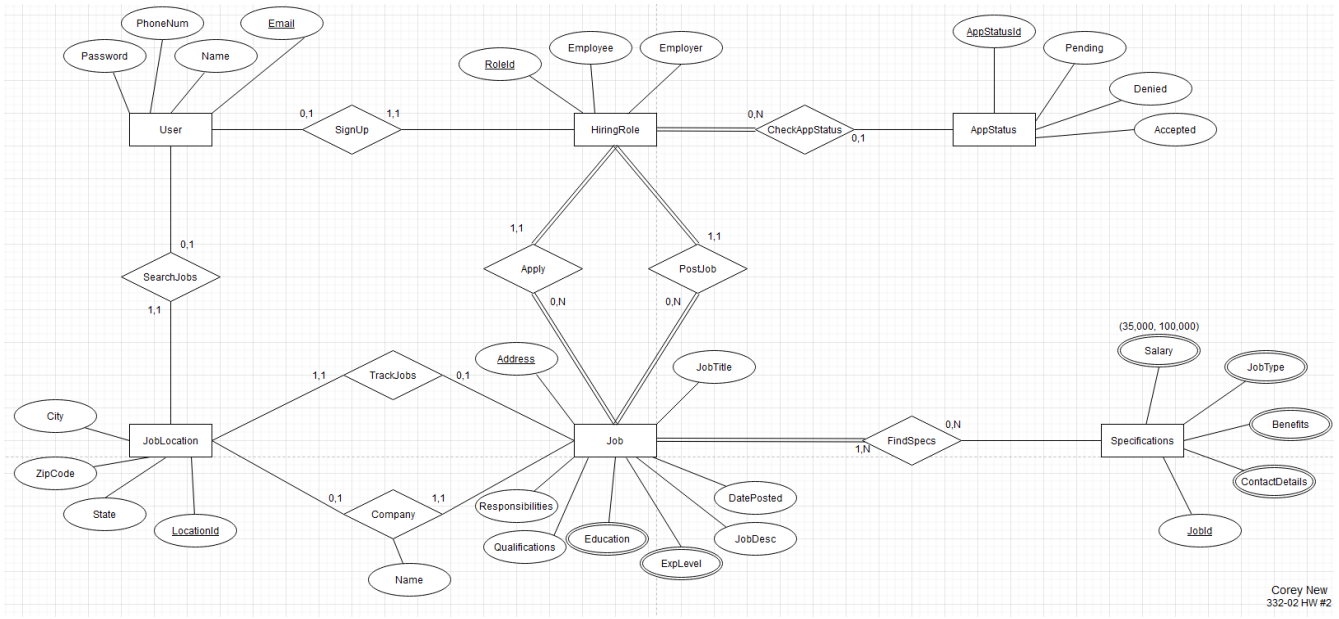
# Josh Lollis



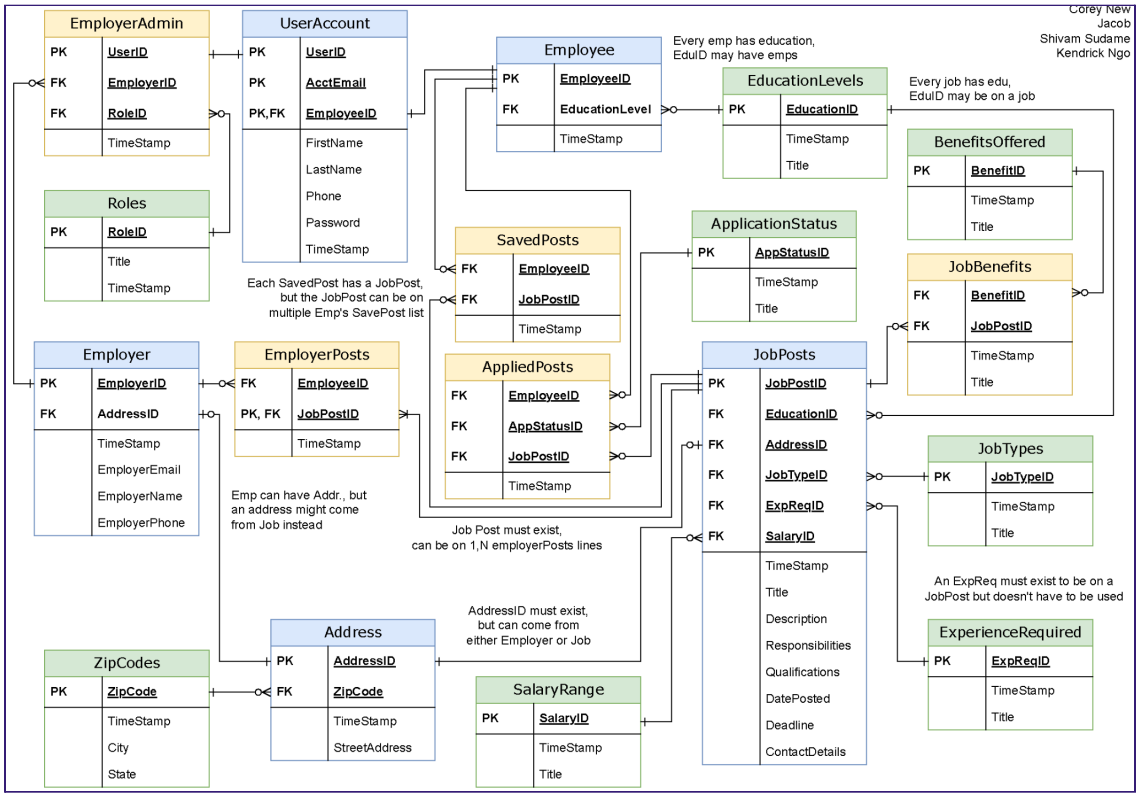

# Gabriel Warkentin



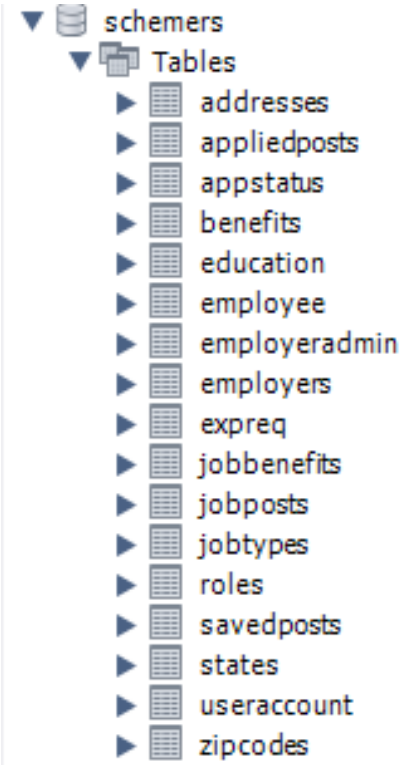
# Corey New




# Relational Model



# Physical Model




# Application Design

## Overview

- The final product website contains php file's to run our front-end UI design with the help of Bootstrap 5.
- Homepage is where the user is able to navigate to different features as listed in the requirements
    - Register an account (if no respective user has already been created)
    - Login to an account
    - Create an Employer account
    - Create an Employee account
    - Post a Job and relative specifications as an Employer
    - View Job Posts as an Employee
    - View All Open Job Posts
    - View an Employer's Job posts

- We ran out of time to complete the functionality for the following. However, our data model does support these and we have some sample data populated
    - Employee's can apply to job posts
    - Employee's can view the status of jobs they've applied for
    - Employers can view and change application status of those that applied
    - Anyone can search by ZipCode or City + State
- Since we were working cross platform and with different system paths (mysql is not added to path by default with xampp), we created a special init.php file which will create all of the tables and views by calling shell scripts from the website itself.

## SQL Queries - https://github.com/TelloVisionGames/Relational-Database-Design

- The queries used to design and populate our database are all held in the files
    - /Relational-Database-Design/SQLScripts/createtables.sql
    - /Relational-Database-Design/SQLScripts/insertalldata.sql
    - /Relational-Database-Design/SQLScripts/createViews.sql
- There are simple SELECT, INSERT, and UPDATE operations using mysqli in PHP throughout the codebase. If any became too complex we made them into views to simplify the queries made on the backend code side.
- And example "complex" view

```sql
DROP VIEW IF EXISTS EmployerGT3PostsView;
CREATE VIEW EmployerGT3PostsView AS
SELECT E.EmployerID, E.EmployerName, E.Email AS EmployerEmail,
       U.FirstName, U.LastName, U.Email AS UserEmail,
       COUNT(J.JobPostID) AS JobsPosted
FROM Employers AS E
INNER JOIN EmployerAdmin AS EA ON EA.EmployerID = E.EmployerID
INNER JOIN UserAccount AS U ON EA.UserID = U.UserID
INNER JOIN JobPosts AS J ON E.EmployerID = J.EmployerID
GROUP BY E.EmployerID
HAVING COUNT(J.JobPostID) >= 3
ORDER BY E.Time_Stamp DESC;
```

## Summary

Overall, the product our group was able to finish contains a variety of features that adhere to the vital requirements of the application. However, there is room for the extra requirements to make the application more stable. The take away from this is that we were able to execute SQL queries in order to manipulate the created and pre-defined data on our application. We believe our database design is robust and would fit the requirements of the company, we simply needed more time to implement the additional functionality.