CSC 4330 Scrum 3 Report

April 6, 2020

Kareem Abdo

Christopher Chee

Derek Delahoussaye

Taylor Olinde

Quoc Than

**Github:** <https://github.com/tolind3/CSC4330-ROCA-GroupProject>

**Intro:**

Our group has followed the instructions for wanted system requirements and has put together a job search platform for ROCA.sa engineering development association. The platform allows current employees and applicants to browse through position postings and apply for any they qualify for. We used tools such as HTML, CSS, JS, and PHP; mostly PHP. We then hosted on a local server using XAMPP. We created our database using MySQL Workbench to incorporate back-end development. Our final product has a website that requests and retrieves from the database. It is also simplistic for easy-use and has met most of the functionality demanded of the clients

**Overall Architecture**

* Multilayered (Client-Server) Architecture
  + Presentation Layer
  + Data Layer

**Project Backlog**

**Employee**

* As an employee, I can access the system by entering my job identification number on the login website.
* As an employee, I can send messages to other employees by entering their job identification number and the message.
* As an employee, I can request a recommendation letter from my immediate supervisor by entering their job identification message.

**Recruiter**

* As a recruiter, I can enter keywords to filter applications by resume.
* As a recruiter, I can choose to receive notifications from the system about upcoming interviews.
* As a recruiter, I can select an option to display a list of outstanding candidates and the status of their applications.

**Applicant**

* As an applicant, I can upload my resume to the system by filling out the resume form with my information.
* As an applicant, I can create my own profile page by entering my personal information and areas of interest.
* As an applicant, I can sign up to receive email notifications about recent positions and job matches.
* As an applicant, I can only apply to a job once.
* As an applicant, I can perform a job search to see the number of positions available, listing jobs in order of most relevant and newer postings, without postings that have already been filled.
* As an applicant, I can select filters to apply to the job search such as salary, state, and level of education.

**Stakeholder**

* As a stakeholder, I can select an option to generate a list of employees applying to open positions and also display the percentage of them that got a new job.
* As a stakeholder, I can view statistics showing the 5 companies with the most number of applicants, the 5 most applied to companies, and the area/field receiving the highest number of applications.

**System Requirements**

* The database should store information about all applicants, open positions, and internal and external applications.
* The system should only allow an applicant to apply to a specific job once. After an applicant has applied, they will not be able to submit another application.
* The system should prevent applications from being overwritten.
* The system should prevent applications from being submitted after the application deadline.
* The system should only accept up to 100 applications per job. After a job has received 100 applications, applicants will see a notification that the maximum number of applications has been reached when they attempt to apply.
* The system should allow job openings to be marked for current employees only. Non-employees will see a notification that the opening is reserved for current employees only.
* The system should only allow applicants to apply to up to 20 different jobs. Once an applicant has submitted 20 job applications, they will receive a notification that they have reached the maximum and will not be able to submit any more applications until another application has been resolved.
* The system should return job openings in order of relevancy and recency. Jobs postings that have already been filled will be excluded from the search results.

**Table of Responsibilities**

|  |  |
| --- | --- |
| Database Management | Taylor Olinde, Kareem Abdo |
| Development (Front-end) | Christopher Chee, Derek Delahoussaye (JS), Quoc Ai Than |
| Github | Everyone |
| Project Deliverables and Reports | Everyone |
| Use Case Diagram | Derek |
| Sequence Diagram | Chris |
| Class Diagram/Database Schema | Taylor |
| Set up MySQL Workbench database | Derek |

**Sprint 1 Deliverables:**

* Have a strategy for implementing front-end development code
* Consult with project manager on proper system requirements

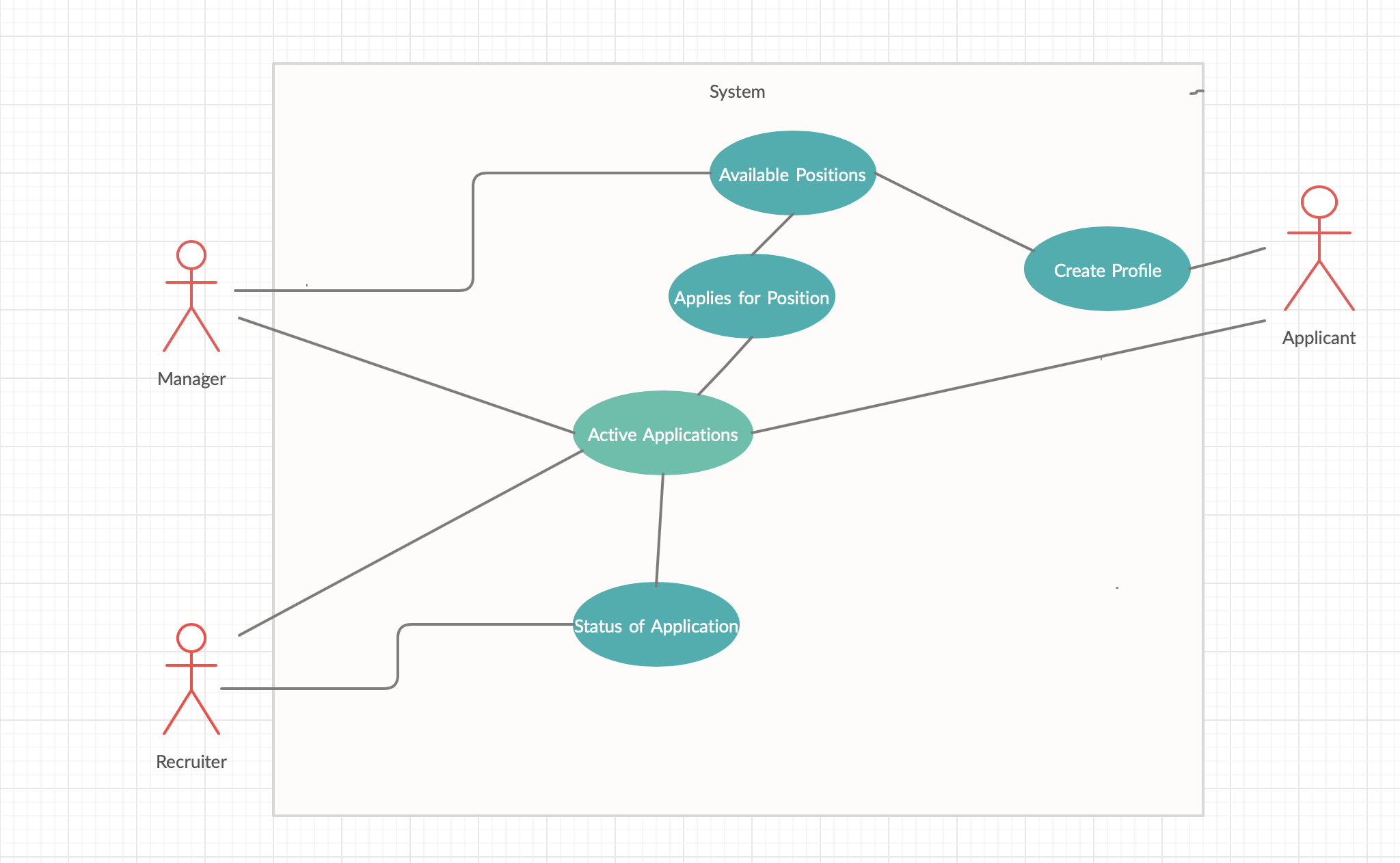
**Sprint 2 Deliverables:**

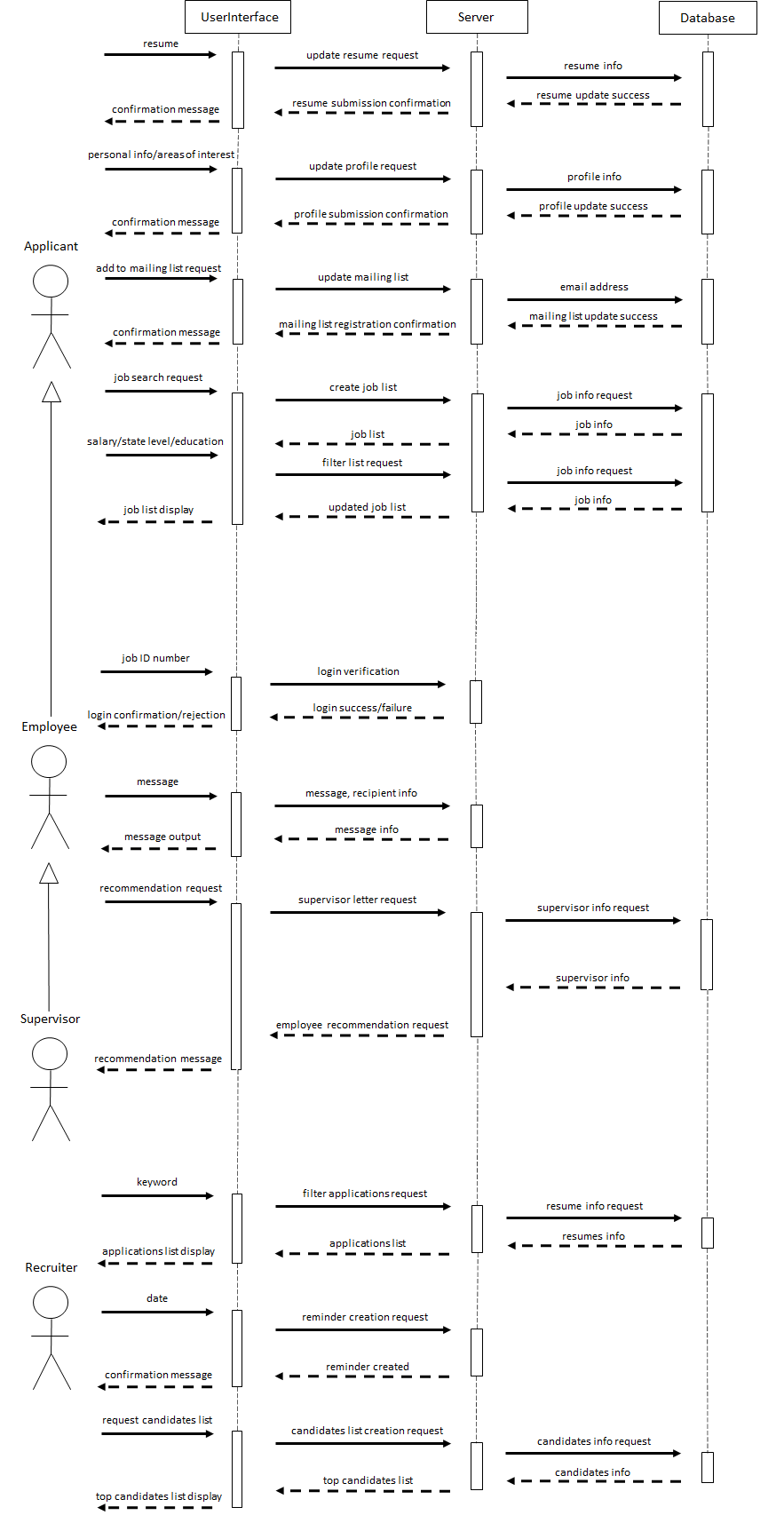
* Have a strategy for implementing front-end development code
* Consult with project manager on proper system requirements
* Setup database to keep track of active applications or status of applications.
* Setup database and create a webpage to display / create a user profile for applicants with information such as email address and areas of interest.
* Setup database that keeps track of applicants, open positions, and other internal or external applications.
* Write SQL queries to test database features

**Sprint 3 Deliverables:**

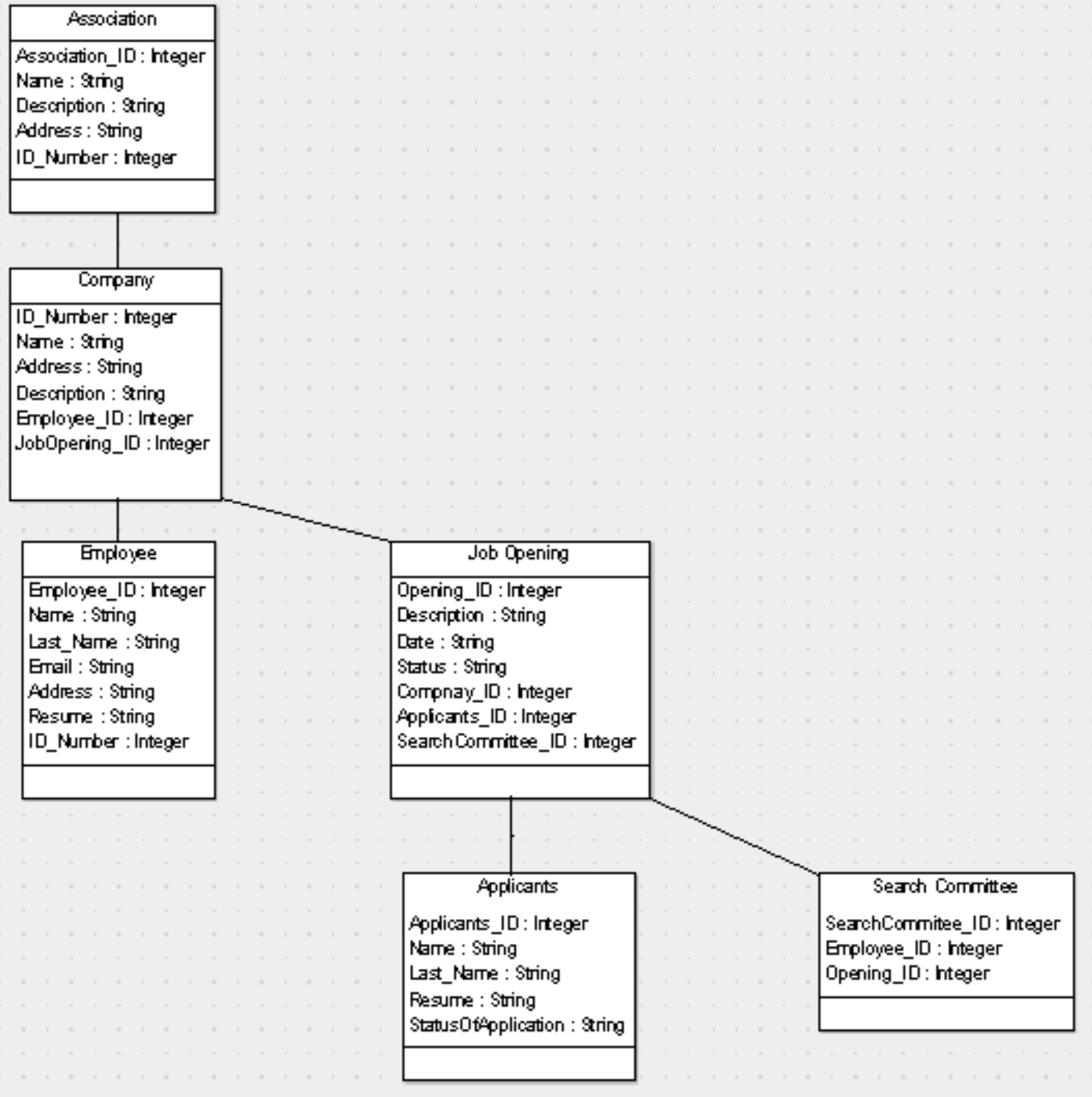
* The database should store information about all applicants, open positions, and internal and external applications.
* The system should only allow an applicant to apply to a specific job once. After an applicant has applied, they will not be able to submit another application.
* The system should prevent applications from being overwritten.
* The system should prevent applications from being submitted after the application deadline.
* The system should only accept up to 100 applications per job. After a job has received 100 applications, applicants will see a notification that the maximum number of applications has been reached when they attempt to apply.
* The system should allow job openings to be marked for current employees only. Non-employees will see a notification that the opening is reserved for current employees only.
* The system should only allow applicants to apply to up to 20 different jobs. Once an applicant has submitted 20 job applications, they will receive a notification that they have reached the maximum and will not be able to submit any more applications until another application has been resolved.
* The system should return job openings in order of relevancy and recency. Jobs postings that have already been filled will be excluded from the search results.

**Use Case Diagram**



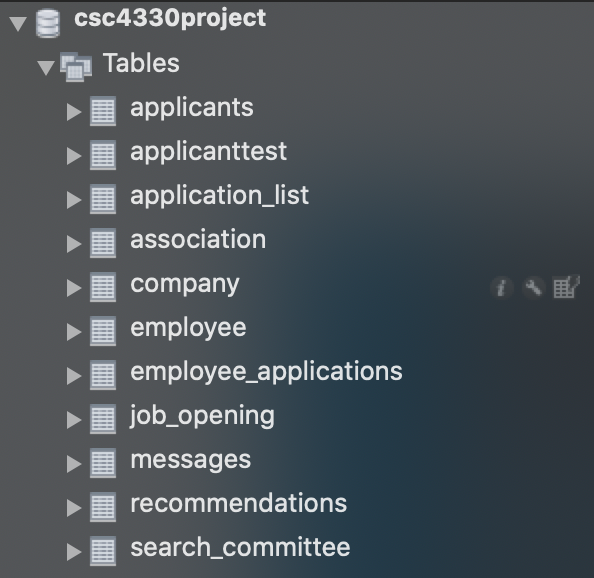
**Sequence Diagram**

**Class Diagram**



**Database Info**

The database is a MySQL database set up via Microsoft Azure. Tables are shown below. Tables are



**Database Table Creation Statements**

CREATE DATABASE csc4330project;

USE csc4330project;

CREATE TABLE IF NOT EXISTS association(

association\_id INT NOT NULL,

name VARCHAR(30) NOT NULL,

description VARCHAR(100),

address VARCHAR(50),

id\_number INT NOT NULL,

CONSTRAINT association\_pk PRIMARY KEY (association\_id)

);

CREATE TABLE IF NOT EXISTS company(

id\_number INT NOT NULL,

name VARCHAR(30) NOT NULL,

address VARCHAR(50),

description VARCHAR(100),

employee\_id INT NOT NULL,

jobOpening\_id INT NOT NULL,

CONSTRAINT company\_pk PRIMARY KEY (id\_number)

);

CREATE TABLE IF NOT EXISTS employee(

employee\_id INT NOT NULL,

name VARCHAR(15),

last\_name VARCHAR(15),

email VARCHAR(30),

address VARCHAR(50),

resume VARCHAR(1000),

id\_number INT NOT NULL,

CONSTRAINT employee\_pk PRIMARY KEY (employee\_id)

);

CREATE TABLE IF NOT EXISTS job\_opening(

opening\_id INT NOT NULL,

description VARCHAR(100),

date VARCHAR(15),

status VARCHAR(15),

company\_id INT NOT NULL,

applicants\_id INT NOT NULL,

searchCommittee\_id INT NOT NULL,

CONSTRAINT job\_opening\_pk PRIMARY KEY (opening\_id)

);

CREATE TABLE IF NOT EXISTS applicants(

applicants\_id INT NOT NULL,

name VARCHAR(15),

last\_name VARCHAR(15),

resume VARCHAR(1000),

statusOfApplication VARCHAR(15),

CONSTRAINT applicants\_pk PRIMARY KEY (applicants\_id)

);

CREATE TABLE IF NOT EXISTS search\_committee(

searchCommittee\_id INT NOT NULL,

employee\_id INT NOT NULL,

opening\_id INT NOT NULL,

CONSTRAINT search\_committee\_pk PRIMARY KEY (searchCommittee\_id)

);

ALTER TABLE association ADD CONSTRAINT association\_company\_fk FOREIGN KEY (id\_number) REFERENCES company (id\_number);

ALTER TABLE company ADD CONSTRAINT company\_employee\_fk FOREIGN KEY (employee\_id) REFERENCES employee (employee\_id);

ALTER TABLE company ADD CONSTRAINT company\_job\_opening\_fk FOREIGN KEY (jobOpening\_id) REFERENCES job\_opening (opening\_id);

ALTER TABLE employee ADD CONSTRAINT employee\_company\_fk FOREIGN KEY (id\_number) REFERENCES company (id\_number);

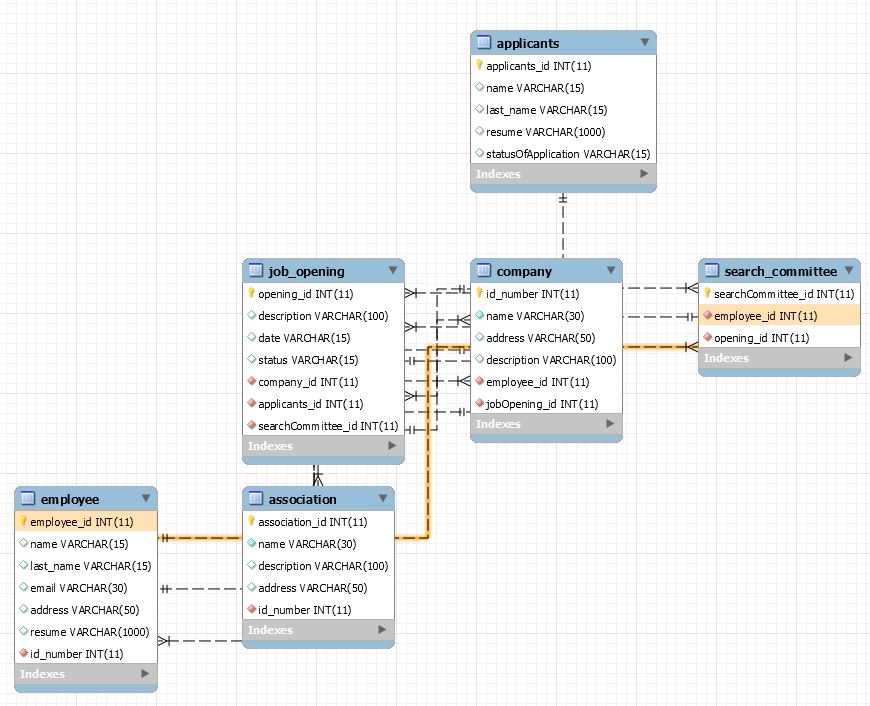
ALTER TABLE job\_opening ADD CONSTRAINT job\_opening\_company\_fk FOREIGN KEY (company\_id) REFERENCES company (id\_number);

ALTER TABLE job\_opening ADD CONSTRAINT job\_opening\_applicants\_fk FOREIGN KEY (applicants\_id) REFERENCES applicants (applicants\_id);

ALTER TABLE job\_opening ADD CONSTRAINT job\_opening\_search\_committee\_fk FOREIGN KEY (searchCommittee\_id) REFERENCES search\_committee (searchCommittee\_id);

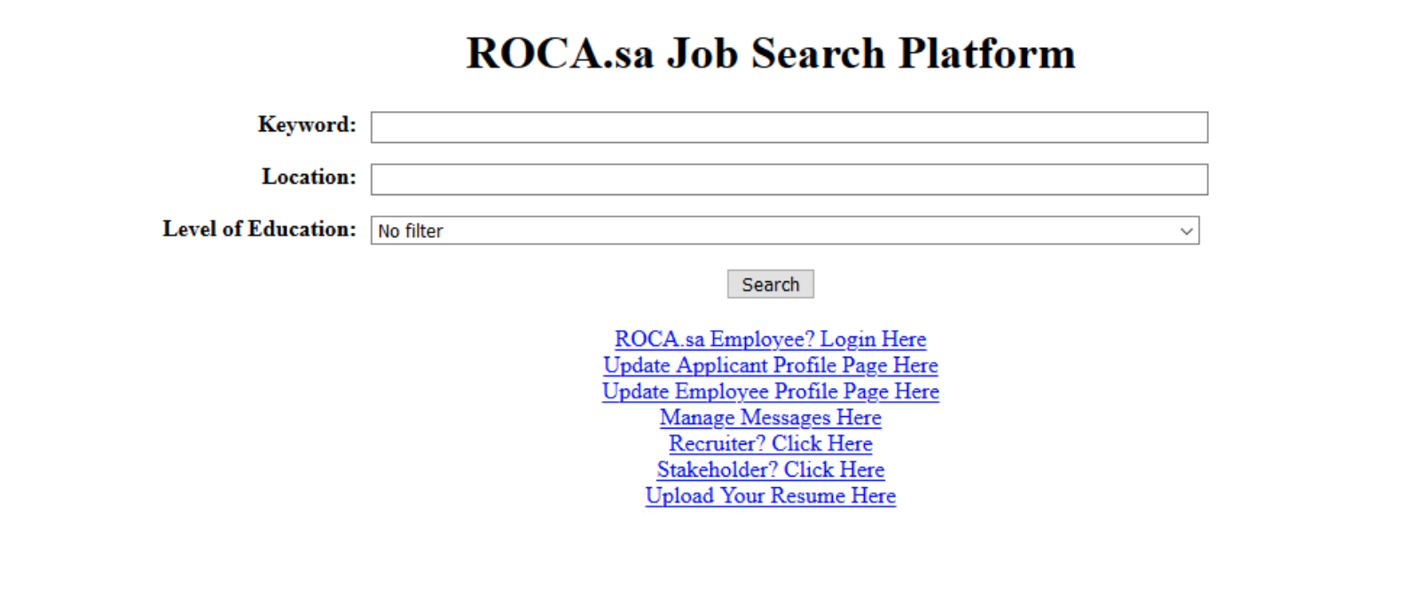
ALTER TABLE search\_committee ADD CONSTRAINT search\_committee\_employee\_fk FOREIGN KEY (employee\_id) REFERENCES employee (employee\_id);

ALTER TABLE search\_committee ADD CONSTRAINT search\_committee\_job\_opening\_fk FOREIGN KEY (opening\_id) REFERENCES job\_opening (opening\_id);

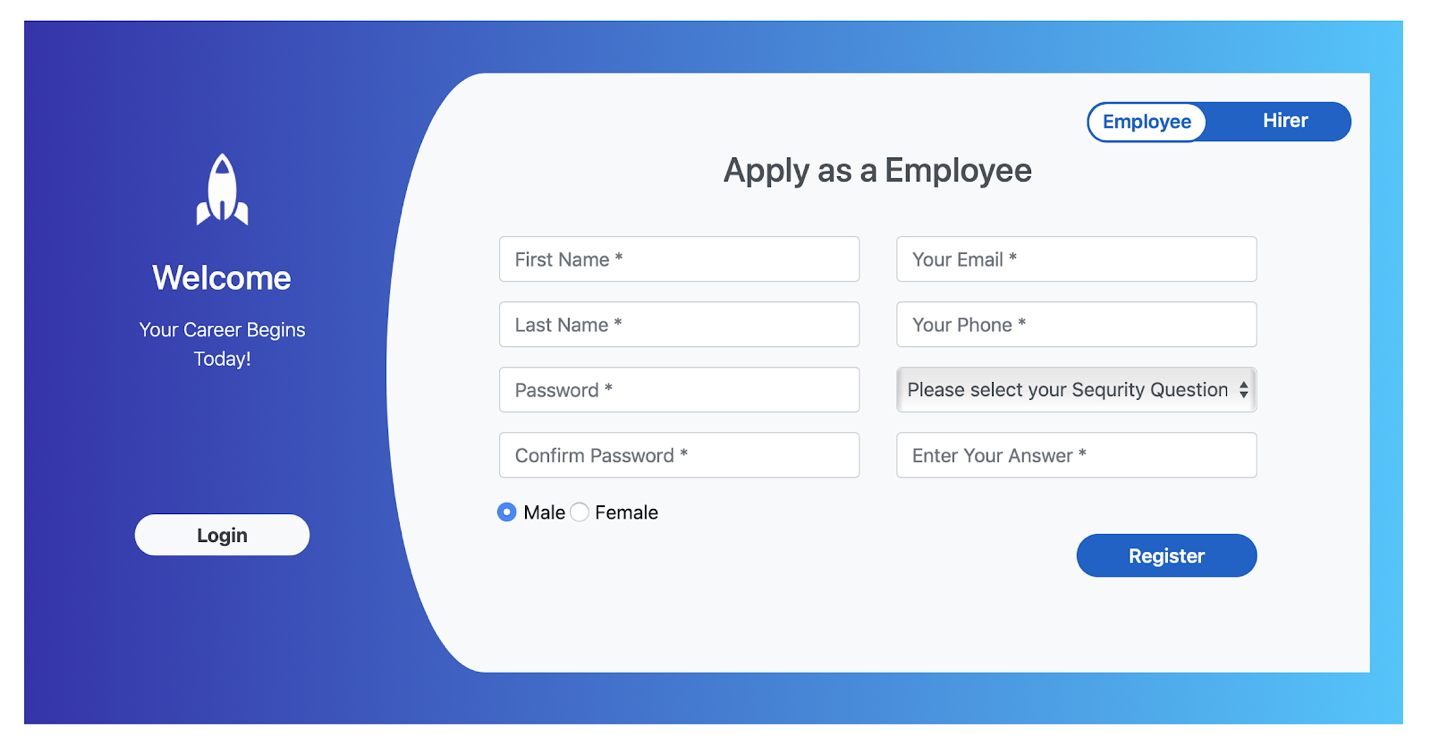
**ER Diagram**

**Webpages**

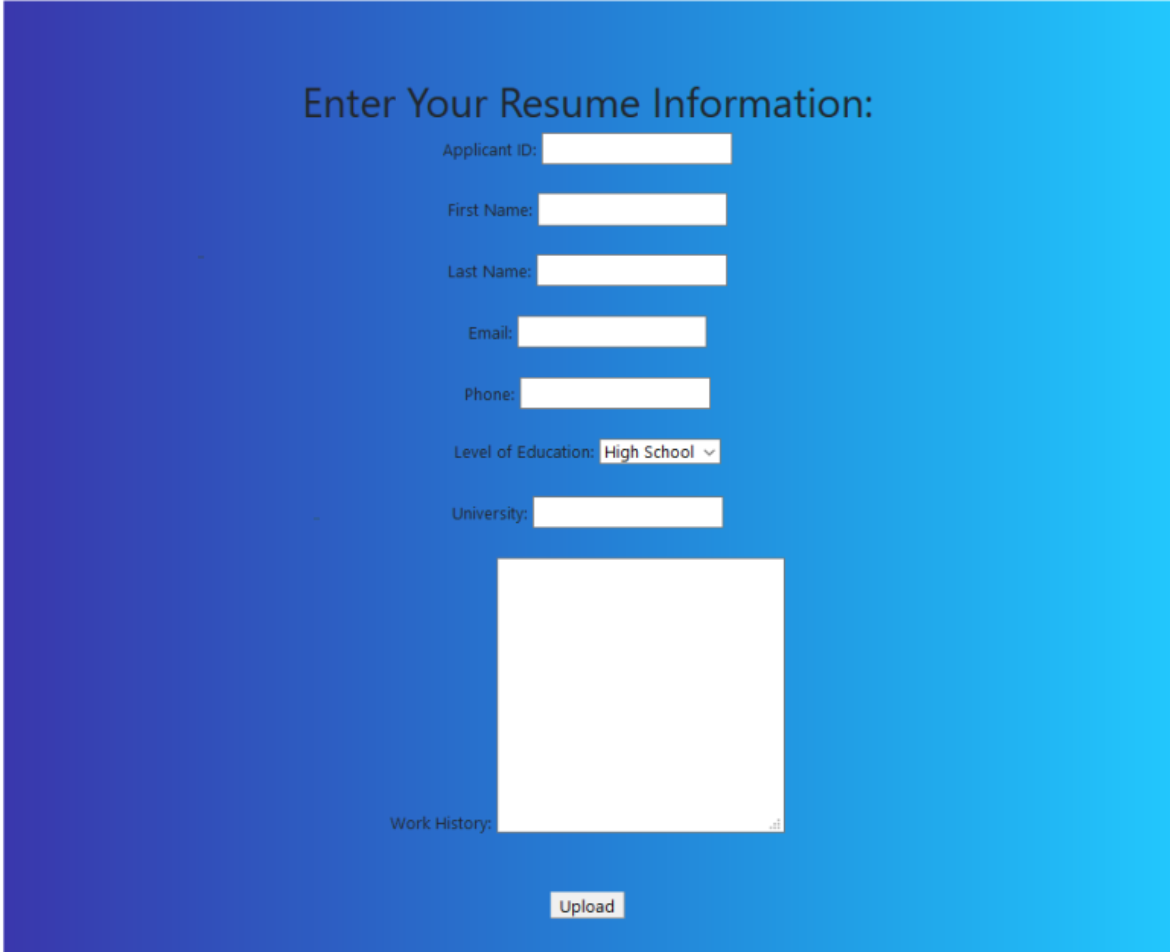
**Main Page**

****

**Employee Login Page**



**Resume Upload Page**

****

**Messaging System Page**

