



**Birla Institute of Technology & Science, Pilani**  
Dubai Campus

Semester II 2021-2022  
Database Systems (CS F212)

## **Recruitment Database Management System**



**Submitted By:**

2020A7PS0080U Bharathwaj M  
2020A7PS0107U Ishan Anthony Menezes  
2020A7PS0128U Anurag Kumar Jha  
2020A7PS0198U Ahmed Thahir  
2020A7PS0221U Anmol Deepak Kumar

**Date of Submission:** 06/04/2022

**Submitted To:** Ms. Sapna Sadhwani

## Summary

| ID            | Name                  | Contribution            |
|---------------|-----------------------|-------------------------|
| 2020A7PS0080U | Bharathwaj M          | Queries                 |
| 2020A7PS0107U | Ishan Anthony Menezes | Queries, RA, RC         |
| 2020A7PS0128U | Anurag Kumar Jha      | Application Backend     |
| 2020A7PS0198U | Ahmed Thahir          | Diagrams, Documentation |
| 2020A7PS0221U | Anmol Deepak Kumar    | Normalization, Report   |

### Title

Recruitment Database Management System

### Repository

The code and the documentation for this project is available on this [Github Repo](#).

### Concepts Used

- ☒ ER Model
- ☒ ER Diagram
- ☒ Relational Model
- ☒ Schema Diagram
- ☒ Tuple Relational Calculus
- ☒ Domain Relational Calculus
- ☒ Normalization (1NF, 2NF, 3NF)
- ☒ mySQL
  - ☒ Queries
  - ☒ Subqueries
  - ☒ Joins
  - ☒ Functions
  - ☒ Procedures
  - ☒ Triggers
- ☒ GUI implementation

## Acknowledgement

Firstly, we would like to thank our family and friends for their constant support throughout our academic life and for this project in particular. They are who inspire us to keep marching forward in our academic journey.

Secondly, we would like to express our sincere gratitude to all Department of Computer Science Professors for allowing us to apply our expertise in this assignment. This project has been a very good opportunity to test our skills.

Thirdly, we would like to thank Mr. Ahamed Jameel, who proposed this idea to help out his (and Thahir's) hometown get better employment opportunities in the UAE. His guidance and ideas throughout this project have been insightful and valuable. We hope that this project turns out to be fruitful and helps out a lot of people.

Furthermore, we would like to thank our university BITS Pilani, Dubai Campus for providing us with the required infrastructure to facilitate efficient learning, especially at the time of a [declining] pandemic.

Lastly, our humble and noteworthy appreciation is due to Ms. Sapna Sadhwani, Dr. Tamizharasan Periyaswamy and Dr. Pramod Gaur for their guidance and assistance in the completion of this assignment. We are also grateful to all the Department of Computer Science faculty for giving us the required knowledge to program using the MySQL querying language to implement our ideas for this project.

## Table of Contents

### Summary

- Title
- Repository
- Concepts Used

### Acknowledgement

### Introduction

- Historical Background
- Theory
  - Database
  - Relational Database
  - Transaction
  - DBMS
  - Querying Language
  - Database Design
  - ER Model
  - ER Diagram
  - Relational Model
  - Schema Diagram
  - Normalization
- Tools
- Limitations

### ER Diagram

### Normalization

- Functional Dependencies
- 1NF
- 3NF

### Schema Diagram

### Design

### Implementation

- Connections
- PHP File Roles
- Inbuilt Queries

  - addcompany.php

  - adduser.php

  - apply.php

  - checkcompanylogin.php

  - checklogin.php

  - city.php

  - index.php

  - jobpagination.php

  - search.php

  - view-job-post.php

### Screenshots

### Other Queries

- Retrieve company number from name
- Retrieve qualifying jobs
- Retrieve jobs in a city
- Retrieve user details if applied to a certain job

Retrieve qualified jobs greater than some salary  
Retrieve users with applications sent out  
Retrieve cities with jobs available  
Check if company email exists  
Check if user email exists  
Retrieve jobpost details from matching jobpost  
Retrieve company details  
Retrieve applications that user has applied for  
Retrieve cities in a state  
Retrieve certain jobpost and their company details  
Triggers

## **Usage**

## **References**

# Introduction

## Historical Background

The job industry is booming with millions of job seekers each years. One of the most crucial elements which the job industry depends upon is the Database Management System, which is used to store, organize and retrieve the details of applicants. This requires a robust and efficient design of the Database to manage the data so that it can be accessed, modified, and stored quickly, with minimal storage space.

Thahir's hometown Kayalpatnam currently has a large number of job seekers in the UAE. We would like to use this application to help them reach potential employers.

## Theory

The following is the theory relevant to this project.

### **Database**

A database is a collection of data.

### **Relational Database**

A database is a collection of data, in the form of rows and columns.

### **Transaction**

A transaction is an atomic unit of a database operation. Hundreds of transactions may take place concurrently at any instant. A transaction may be one of the CRUD operations

- Create
- Read
- Update (or Insert)
- Delete

### **DBMS**

A 'DataBase Management System' is a system for handling large amounts of data stored in one or more databases. It is also called as a 'Transaction Processing System'.

### **Querying Language**

A querying language is a set of commands that facilitates database operations.

### **Database Design**

There are various ways of designing the structure(schema) of a database. The database must be designed well before inserting values, to ensure inconsistencies.

The logical schema depicts the structure of the database, showing the tables, columns, and relationships with other tables in the database, and is a direct mapping of the Entity-Relationship diagram. The physical schema is created by actually generating the tables, columns, and relationships in the DBMS, using a querying language.

Relational databases are ACID-compliant, but NoSQL aren't

- Atomicity  
If any sub-steps of a transaction fails, the whole transaction must fail and the database must be in the same state as the original.
- Consistency  
Correct data is ensured through constraints
- Isolation  
Concurrent-Execution Safe  
Simultaneous transactions must be considered as multiple sequential transactions.
- Durability  
Committed transactions must be stored to a non-volatile memory, to prevent loss of data.

### **ER Model**

A database is modelled as

- entity set
- relationship sets

| Term | Meaning | Example |
|------|---------|---------|
|------|---------|---------|

| Term                       | Meaning  | Example                                      |
|----------------------------|--|--|
| Entity                     | unique object  | specific person, company                     |
| Entity Set                 | set of entities  |  |
| Attributes                 | properties/features of an entity/relationship            | name, age                                    |
| Composite Attributes       | sub-attributes   | first name, last name                        |
| Relationship               | association among several entites                        |  |
| Relationship Sets          | set of relationships                                     |  |
| Degree of Relationship Set | no of entity sets that participate in a relationship set |  |
| Mapping Cardinalities      | Type of mapping  | One-One<br>One-Many<br>Many-One<br>Many-Many |

## ER Diagram

A diagrammatic representation of the entities and relations used.

| Shape            | Meaning   |
|------------------|---|
| Rectangle        | Entity Set  |
| Double Rectangle | Weak Entity Set<br>entity without a primary key                                       |
| Diamond          | Relationship Set  |
| Double Diamond   | Weak Relationship Set<br>relation connecting a weak entity with something else        |
| Ellipse          | Attribute   |
| Dashed ellipse   | Derived attribute   |
| Double ellipse   | Multi-valued attribute  |
| Underline        | Primary Key Attribute   |
| Triangle         | 'is-a' relation   |
| Lines            | - Link attribute <b>to</b> entity set<br>- Link entity set <b>to</b> relationship set |
| →                | one   |
| —                | many  |

## Relational Model

A database is a collection of relations, where each relations is a table.

## Schema Diagram

A diagrammatic representation of the relations used.

## Normalization

It is the process of structuring a database, usually a relational database, in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity.

|                               | 1NF | 2NF | 3NF | BCNF | 4NF | 5NF |
|-------------------------------|-----|-----|-----|------|-----|-----|
| No Multi-Valued Attributes    | ✓   | ✓   | ✓   | ✓    | ✓   | ✓   |
| No Partial Dependency         |     | ✓   | ✓   | ✓    | ✓   | ✓   |
| No Transitive Dependency      |     |     | ✓   | ✓    | ✓   | ✓   |
| LHS = Candidate/Super Key     |     |     |     | ✓    | ✓   | ✓   |
| No Multi-Attribute Dependency |     |     |     |      | ✓   | ✓   |
| Lossless Decomposition        |     |     |     |      |     | ✓   |

## Tools

| Tool      | Full Form                                   | Use   |
|-----------|---|---|
| HTML      | HyperText Markup Language                   | Layout of web pages   |
| CSS       | Cascading Style Sheets                      | Design of web pages   |
| Bootstrap |   | CSS framework for developing responsive web pages             |
| Php       | Hypertext Preprocessor                      | Backend<br>- Calling SQL<br>- Serving dynamic content to user |
| mySQL     | My Structured Query Language                | Structured Relational Database<br>Querying Language           |
| Xampp     | cross-platform, Apache, MySQL, PHP and Perl | - Database manipulation<br>- Locally hosting the website      |

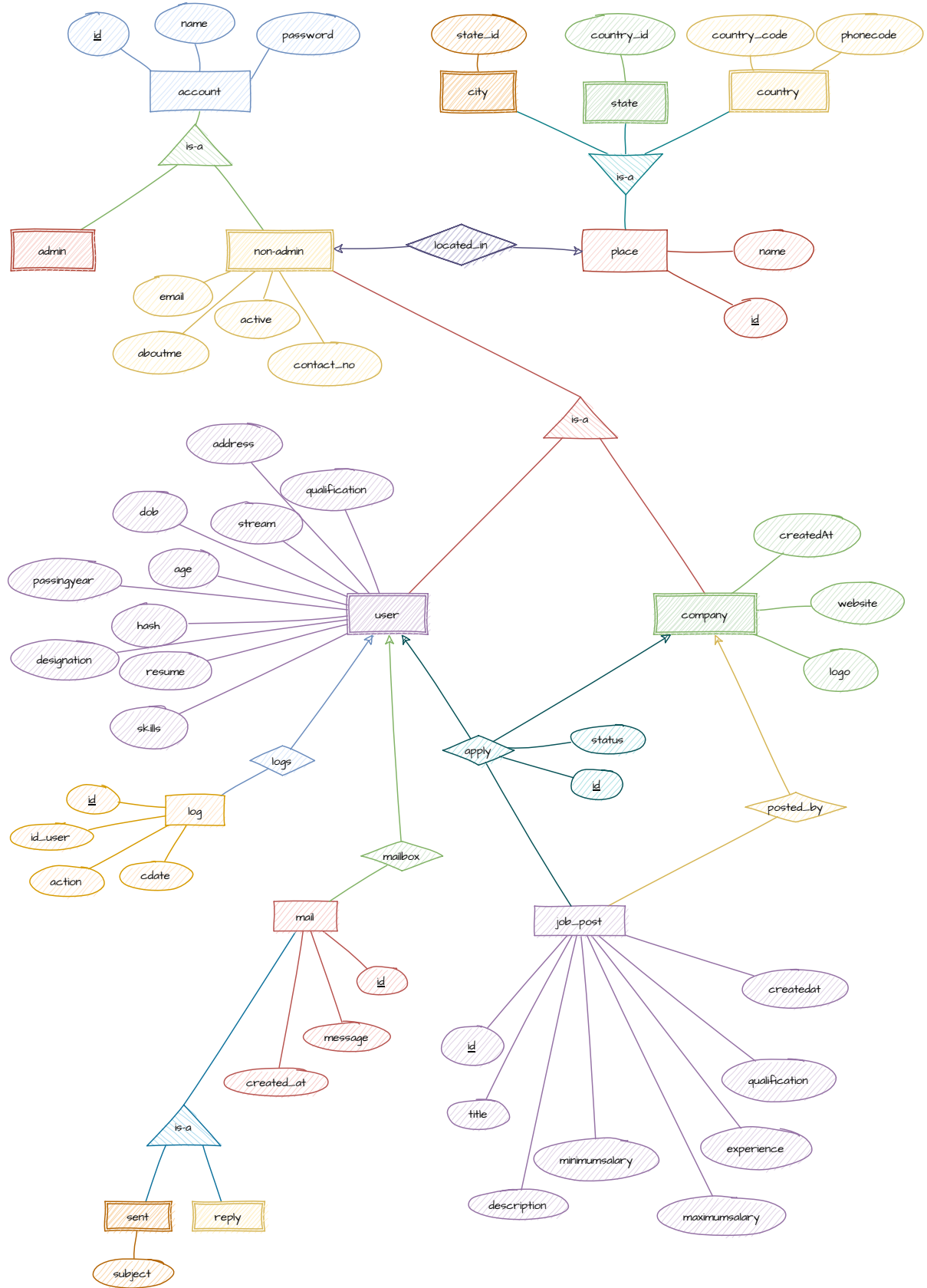
## Limitations

The following are the drawbacks of our project. We plan on addressing these issues in the future.

1. Dynamic Sites are not as performant as static sites
2. Filters are not dynamic currently (it is hard-coded)
3. There are a few hyperlinks which don't work properly.  
For example, report button down as the site is not hosted on any server
4. The webpage is hosted locally at the moment. Needs to be run on the server
5. Relational database paradigm is not the most efficient to deal with a networking model, with unstructured data. A NoSQL paradigm could be used in the future to deal with large loads.



# ER Diagram



## Normalization

In this project, we have only reached till 3NF. This is because BCNF and onwards may lead to lossy decomposition, with respect to the functional dependencies.

### Functional Dependencies

admin → id\_admin, username, password

apply\_job\_post → id\_apply, id\_jobpost, id\_company, id\_user, status

cities → id, name, state\_id

company → id\_company, name, companyname, country, state, city, contactno, website, email, password, aboutme, logo, createdAt, active

countries → id, country\_code, name, phonecode

job\_post → id\_jobpost, id\_company, jobtitle, description, minimumsalary, maximumsalary, experience, qualification, createdat

mailbox → id\_mailbox, id\_fromuser, fromuser, id\_touser, subject, message, createdAt

reply\_mailbox → id\_reply, id\_mailbox, id\_user, usertype, message, createdAt

states → id, name, country\_id

users → id\_user, firstname, lastname, email, password, address, city, state, contactno, qualification, stream, passingyear, dob, age, designation, resume, hash, active, aboutme, skills

### 1NF

R(id\_admin, username, password, id\_apply, id\_jobpost, id\_company, id\_user, status, id, name, state\_id, id\_company, name, companyname, country, state, city, contactno, website, email, password, aboutme, logo, createdat, active, id, country\_code, name, phonecode, id\_jobpost, id\_company, jobtitle, description, minimumsalary, maximumsalary, experience, qualification, createdat, id\_mailbox, id\_fromuser, fromuser, id\_touser, subject, message, createdAt, id\_reply, id\_mailbox, id\_user, usertype, message, createdAt, id, name, country\_id, id\_user, firstname, lastname, email, password, address, city, state, contactno, qualification, stream, passingyear, dob, age, designation, resume, hash, active, aboutme, skills)

### 3NF

R1(id\_admin, username, password)

R2(id\_apply, id\_jobpost, id\_company, id\_user,status)

R3(id, name, state\_id)

R4(id\_company, name, companyname, country, state, city, contactno, website, email, password, aboutme, logo, createdat, active)

R5(id, country\_code, name, phonecode)

R6(id\_jobpost, id\_company, jobtitle, description, minimumsalary, maximumsalary, experience, qualification, createdat)

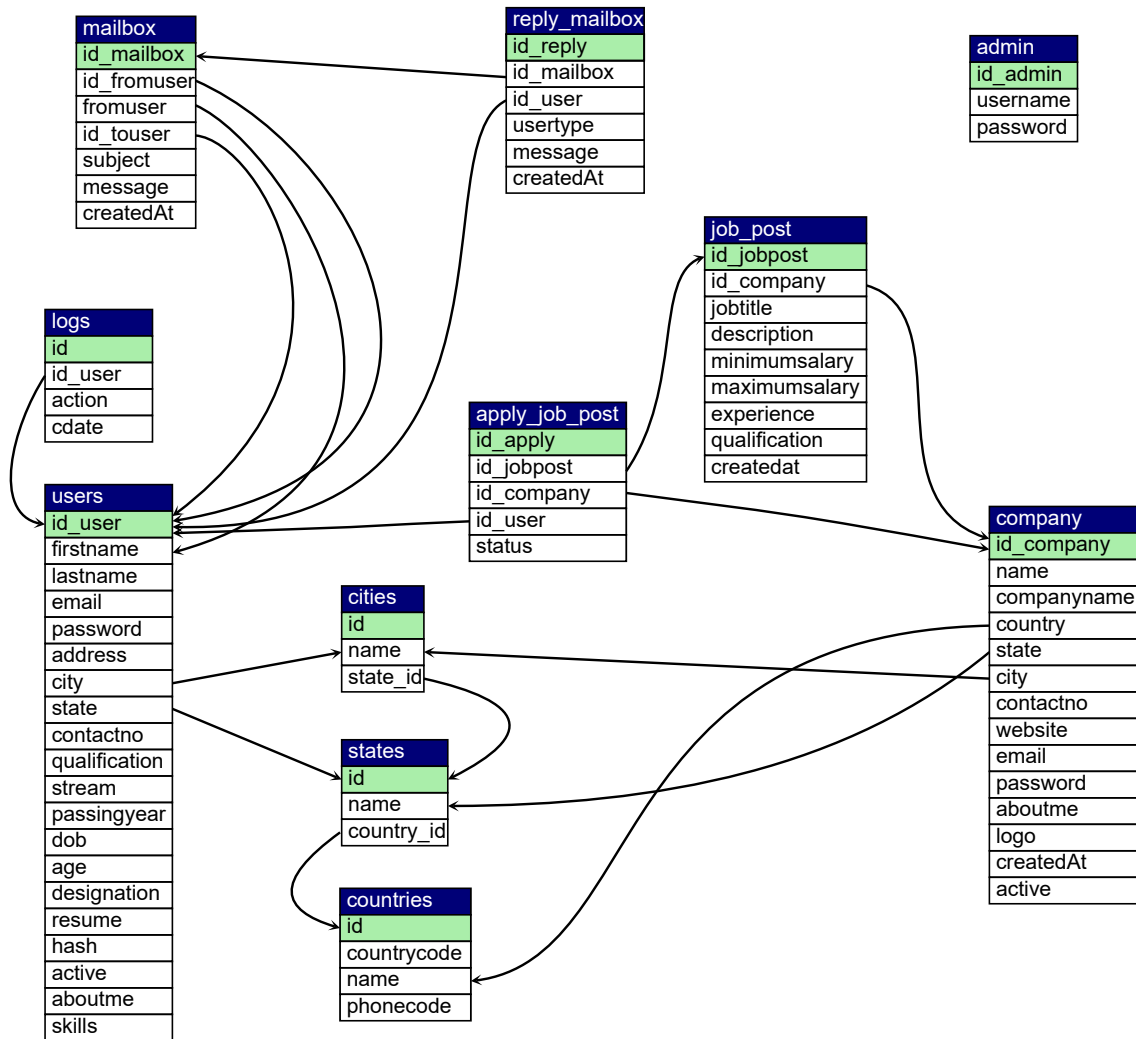
R7(id\_mailbox, id\_fromuser, fromuser, id\_touser, subject, message, createdat)

R8(id\_reply, id \_mailbox, id\_user, usertype, message, createdAt)

R9(id, name, country\_id)

R10(id\_user, firstname, lastname, email, password, address, city, state, contactno, qualification, stream, passingyear, dob, age, designation, resume, hash, active, aboutme, skills)

# Schema Diagram



## Design

| Table                    | Details  |
|--------------------------|--|
| admin                    | <ul style="list-style-type: none"> <li>- admin username</li> <li>- admin password</li> <li>- Holds details about the active companies registered, candidates registered, total job posts and total applications.</li> <li>- Permissions to remove any user or candidate</li> <li>- Any company before being able to recruit and post jobs must be verified by the admin.</li> </ul>  |
| job_post table           | <ul style="list-style-type: none"> <li>- details of jobs posted by a company.</li> <li>- contains the id of the company which posted the job, the job title and description.</li> <li>- contains minimum salary , maximum salary, and qualifications for filtering out candidates.</li> <li>- createdat attribute is used for showcasing the latest 4 jobs in the home page.</li> </ul>  |
| apply_job_post           | <p>Id_apply: acts as primary key with auto increment on</p> <p>Id_job_post: acts as the id for the GET method of php for opening the specific webpage having job description</p> <p>Id_company/user : id of user or the company</p> <p>Status: status of the job post depending on the admin. A job post is published only after validation by the admin.</p>  |
| City<br>state<br>country | <ul style="list-style-type: none"> <li>- Large database of the cities, states and tables connected together by id of state and countries.</li> <li>- Created separately since while creating forms the option to select country comes first, then the state and then the city.</li> <li>- None of these are compulsory so the 3 tables are kept separately so that if a user only provides country or country-state it works as well.</li> </ul> |
| Company                  | <ul style="list-style-type: none"> <li>- Consists of all the details of the company. The logo is stored as image format in the uploads folder.</li> <li>- Any company before being to operate must be able to be verified from the admin when their active column changes from 2 to 1.</li> </ul>  |
| Users                    | <ul style="list-style-type: none"> <li>- Similar to the companies these include all the users looking for jobs.</li> <li>- The resume is stored as pdf(size smaller than 5 mb) and is stored in the uploads folder.</li> <li>- The passwords are stored after applying decoding so that even the admin or the superuser can't access the password.</li> </ul>  |
| Logs                     | <ul style="list-style-type: none"> <li>- contains the log of user activity</li> <li>- updated with the activity of user whenever a new user is created, updated or deleted.</li> </ul>   |
| Mailbox<br>Reply         | <ul style="list-style-type: none"> <li>- Inbuilt mailing technique b/w the users and the companies, the user can have conversation with the employee who have interest in their application.</li> <li>- The design is similar to Gmail</li> <li>- all the information is stored in the sql file i.e. the message, from, to and the date.</li> </ul>  |

## Implementation

### Connections

Let's see how a connection with mysql is made.

In all the php pages which require access to database we include the db.php. db.php is a php which is used for establishing the sql connection. Rather than writing all of that separately in all the php files, we have created one db file and included this in all which we require.

For running any SQL query we just write the query similar to that in SQL and then run it using the

connection established.

```
$result = $conn->query($sql);
```

### PHP File Roles

| PHP File  | Role  |
|---|---|
| <code>index</code>  | The main php file that opens up at the start. Contains link to other pages like login and register and also the admin login. Shows 4(using sql limit) recently available job offers at the centre.  |
| <code>jobs</code>   | Contains the job offers 4 per page(again by using limit 4, for this code is written in <code>jobpagination.php</code> ).<br>The filters applied are using the help of search.php which applies the filters of city and experience.<br>Currently the cities are navi-mumbai and Bengaluru but more can be added as required.<br>Clicking on any of these open the <code>view-job-post.php</code> |
| <code>view-job-post</code>  | - also has an id at the end of url required to identified which job post page to open as all have the same format.<br>- This id is sent using the GET method of php.  |
| <code>login</code><br><code>signup</code><br><code>register</code>            | - These are the pages which open the login and signup pages.<br>- The details of username and password are checked using the <code>checklogin.php</code> and <code>checkcompanylogin.php</code> .<br>- All the login and register pages have the standard php code for validation of phone number, email id, pdf size etc.<br>- These can be seen at W3Schools itself.                          |
| Folders of<br><code>user</code><br><code>company</code><br><code>admin</code> | - These contain the php pages to be loaded when the corresponding people are logged in.<br>- This is known by using the session global variable of php to know which of these is set.<br>- The php files within are pretty simple to those mentioned above with the exception that these are a little more specified for the particular kind of user.   |

### Inbuilt Queries

#### `addcompany.php`

Check if the email already exists so that every new company created has a new email.

```
$sql = "SELECT email FROM company WHERE email='$email'";
```

Inserting new data retrieved from the form into the company table

```
$sql = "INSERT INTO company(name, companyname, country, state,
city,
contactno, website, email, password, aboutme, logo) VALUES
('$name',
'$companyname', '$country', '$state', '$city', '$contactno',
'$website',
'$email', '$password', '$aboutme', '$file')";
```

#### **adduser.php**

Similar to the above to check if the file already exists.

```
$sql = "SELECT email FROM users WHERE email='$email'";
```

Inserting data into the user table

```
$sql = "INSERT INTO users(firstname, lastname, email, password,
address, city, state, contactno, qualification, stream,
passingyear, dob, age, designation, resume, hash, aboutme, skills)
VALUES ('$firstname', '$lastname', '$email', '$password',
'$address', '$city', '$state', '$contactno', '$qualification',
'$stream', '$passingyear', '$dob', '$age', '$designation', '$file',
'$hash', '$aboutme', '$skills')";
```

#### **apply.php**

This is not a file directly associated with the front end but called for processing of another file.

Retrieve all the data from the job\_post table where id is taken from the GET function of php.

```
$sql = "SELECT * FROM job_post WHERE id_jobpost='$_GET[id]'";
```

Check if the user has applied to the job post or not.

```
$sql1 = "SELECT * FROM apply_job_post WHERE
id_user='$_SESSION[id_user]' AND
id_jobpost='$row[id_jobpost]'";
```

Whenever a user applies for a job the details need to be added to the apply\_job\_post table.

```
$sql = "INSERT INTO apply_job_post(id_jobpost, id_company, id_user)
VALUES ('$_GET[id]', '$id_company', '$_SESSION[id_user]')";
```

#### **checkcompanylogin.php**

Check if the login is correct or not.

```
$sql = "SELECT id_company, companyname, email, active FROM company
WHERE email='$email' AND password='$password'";
```

#### **checklogin.php**

Check if the login is correct or not.

```
$sql = "SELECT id_company, companyname, email, active FROM company
WHERE email='$email' AND password='$password'";
```

#### city.php

Select the cities from states. Similar is with the case in `state.php` in a layer by layer like fashion.

```
$sql = "SELECT * FROM cities WHERE state_id='$_POST[id]'";
```

#### index.php

Select 4 most recent job offers to display at the main screen.

```
$sql = "SELECT * FROM job_post Order By createdat Limit 4";
```

Other sql commands are the simple select commands to get the count of various elements.

#### jobpagination.php

Choosing 4 job posts at a time.

```
$sql = "SELECT * FROM job_post LIMIT $start_from, $limit";
```

#### search.php

Search specific company on the basis of inputted name.

```
$sql = "SELECT * FROM job_post WHERE jobtitle LIKE '%$search%'  
LIMIT $start_from, $limit";
```

#### view-job-post.php

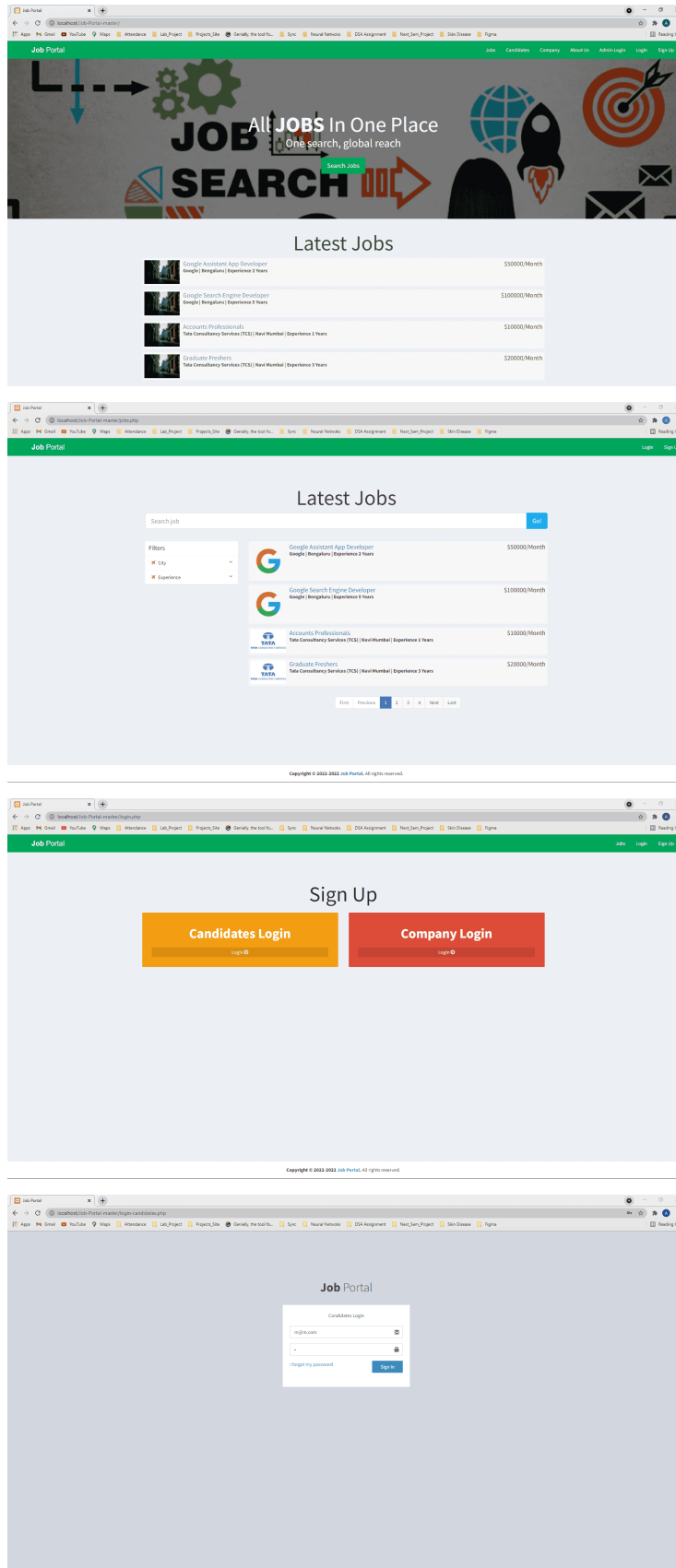
Select details of the specific company on on the page on the basis of id sent by GET method of php.

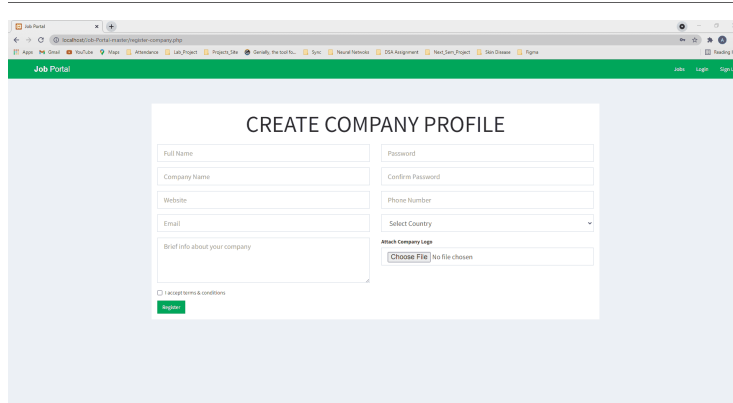
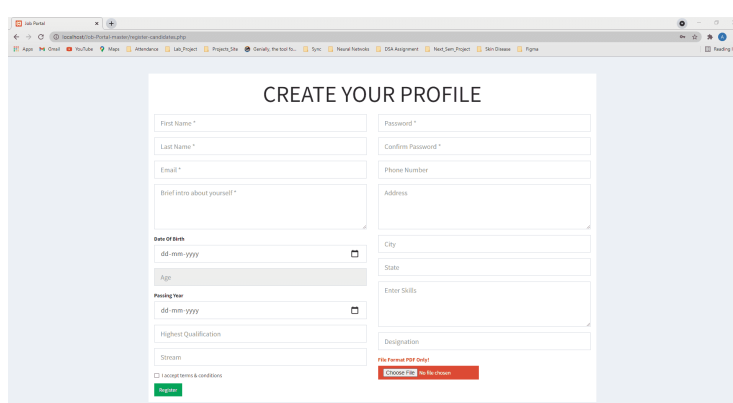
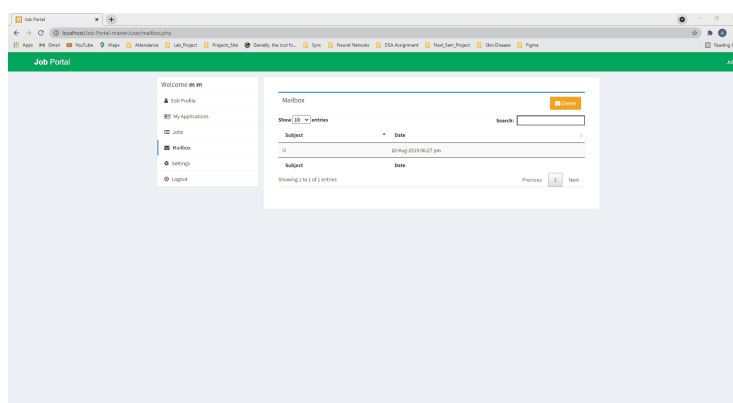
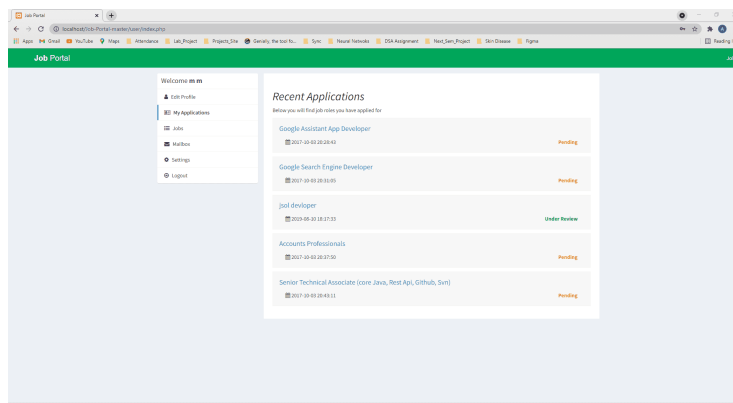
```
$sql = "SELECT * FROM job_post INNER JOIN company ON  
job_post.id_company=company.id_company WHERE  
id_jobpost='$_GET[id]'";
```

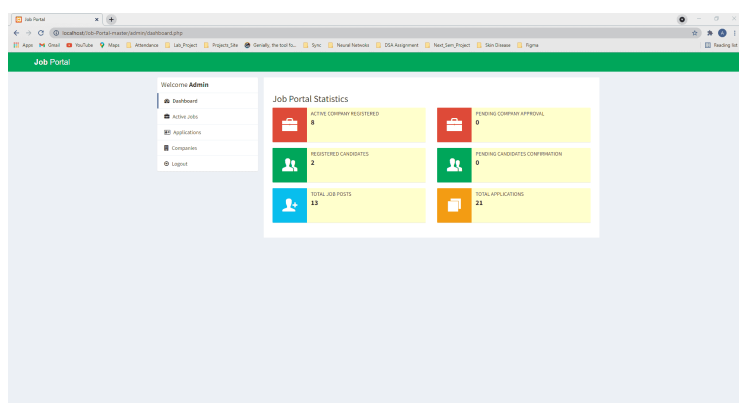
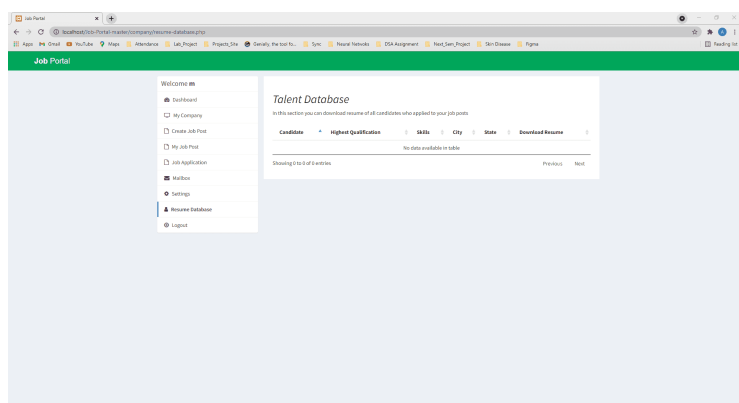
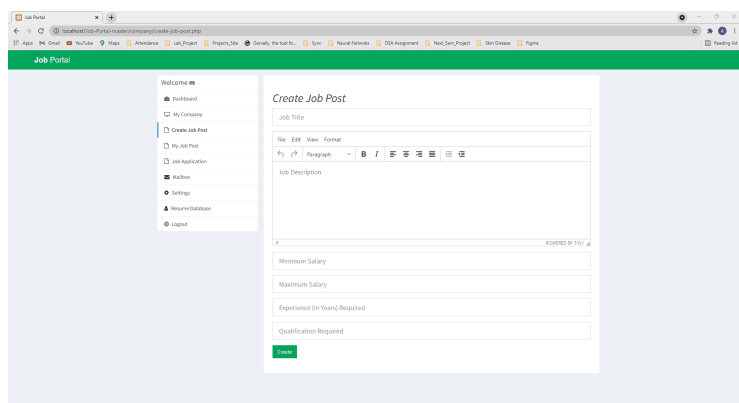
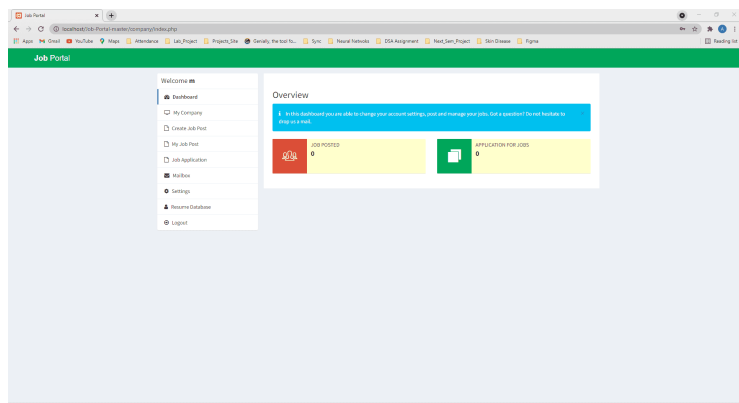
Similar queries in the user, company and admin php files for the specific user types.

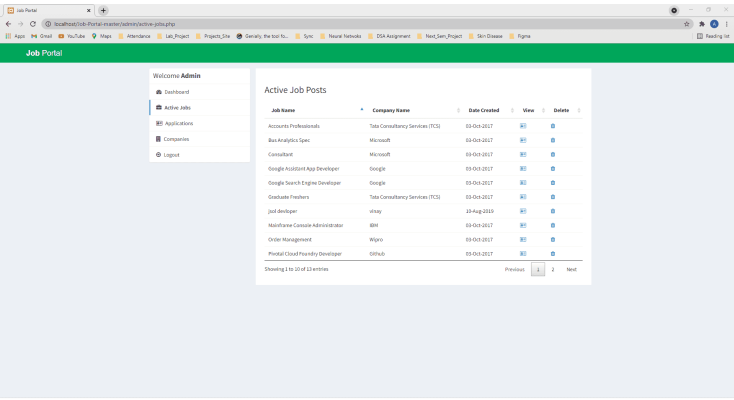


# Screenshots

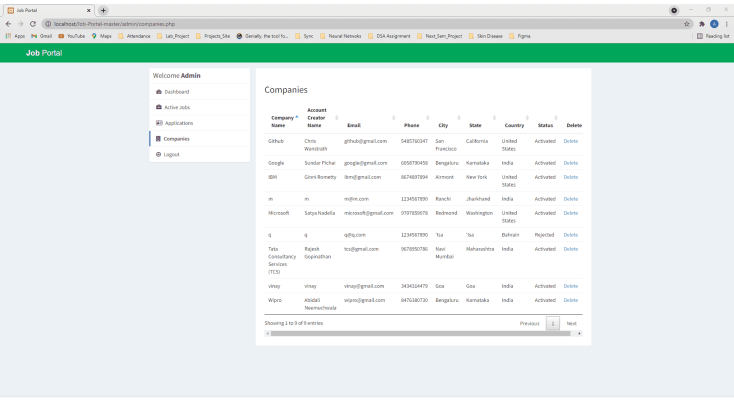








Copyright © 2012-2022 Job Portal. All rights reserved.



Copyright © 2012-2022 Job Portal. All rights reserved.

## Other Queries

### Retrieve company number from name

```
delimiter //
create function contact_from_company(input_companyname
varchar(255))
returns varchar(255)
begin
    return (select contactno
            from company
            where company.companyname= input_companyname);
end //
delimiter ;

select * from contact_from_company(*some company name*);
```

RA

$$\pi_{\text{contactno}}(\sigma_{\text{companyname}='companyname'}(\text{company}))$$

TRC

$$\{t | \exists c \in \text{company} (t[\text{contactno}] = c[\text{contactno}] \wedge c[\text{companyname}] = 'companyname')\}$$

DRC

$$\{ \langle co \rangle | \exists ic, n, cn, c, st, ci, w, e, p, a, l, cd, ac ( \langle ic, n, cn, c, st, ci, co, w, e, p, a, l, cd, ac \rangle \in \text{company} \wedge cn = 'companyname') \}$$

### Retrieve qualifying jobs

```
delimiter //
create procedure job_from_qualification(input_id_user int(11))
begin
    select companyname, jobtitle, description, minimumsalary,
    maximumsalary, experience
    from users
    where users.id_user=input_id_user
    inner join job_post
    on users.qualification=job_post.qualification;
end //
delimiter ;

call job_from_qualification(*some userid*);
```

RA

$$\pi_{\text{jobtitle,description,minimumsalary,maximumsalary,experience}}(\sigma_{\text{id\_user}='userid'}(\text{users} \bowtie \text{job\_post}))$$

TRC

$$\{t | \exists u \in \text{users} (u[\text{id\_user}] = 'userid' \wedge \exists j \in \text{job\_post} ( \\ t[\text{jobtitle}] = j[\text{jobtitle}] \wedge t[\text{description}] = j[\text{description}] \wedge \\ t[\text{minimumsalary}] = j[\text{minimumsalary}] \wedge t[\text{maximumsalary}] = j[\text{maximumsalary}] \wedge \\ t[\text{experience}] = j[\text{experience}] \wedge u[\text{qualification}] = j[\text{qualification}])) \}$$

DRC

$$\{ \langle jt, desc, mins, maxs, ex \rangle \mid \exists iu, fn, ln, em, pa, ad, ci, st, con, q, s, py, dob, age, desg, res, h, act, abt, sk ( \\ \langle iu, fn, ln, em, pa, ad, ci, st, con, q, s, py, dob, age, desg, res, h, act, abt, sk \rangle \in users \wedge iu = 'userid' \wedge \\ \exists ij, ic, crd ( \langle ij, ic, jt, desc, mins, maxs, ex, crd \rangle \in job\_post)) \}$$

### Retrieve jobs in a city

```
SELECT j.companyname, j.jobtitle, j.description, j.minimumsalary,
j.maximumsalary, j.experience
FROM job_post as j, (SELECT id_jobpost, city
FROM job_post, company
WHERE job_post.id_company= company.id_company) as c
WHERE j.id_jobpost=c.id_jobpost AND c.city='city';
```

RA

$$\pi_{companyname, jobtitle, description, minimumsalary, maximumsalary, experience}(\sigma_{city='city'}(job\_post \bowtie company))$$

TRC

$$\{ t \mid \exists j \in job\_post (t[jobtitle] = j[jobtitle] \wedge t[description] = j[description] \wedge \\ t[minimumsalary] = j[minimumsalary] \wedge t[maximumsalary] = j[maximumsalary] \wedge t[experience] = j[experience] \wedge \\ \exists c \in company (j[id\_company] = c[id\_company] \wedge c[city] = 'city')) \}$$

DRC

$$\{ \langle cn, jt, desc, mins, maxs, ex \rangle \mid \exists ij, ic, q, crd ( \langle ij, ic, jt, desc, mins, maxs, ex, crd \rangle \in job\_post \wedge \\ \exists n, c, st, ci, co, w, e, p, ab, l, cd, a ( \langle ic, n, cn, c, s, ci, co, w, e, p, a, l, cd, ac \rangle \in company \wedge ci = 'city')) \}$$

### Retrieve user details if applied to a certain job

```
delimiter //
create procedure resume_from_apply_jobpost(input_id_jobpost
int(11))
begin
select id_user, email, resume
from users, apply_job_post
where apply_job_post.id_jobpost =input_id_jobpost and
apply_jobpost.id_user = users.id_user;
end //
delimiter ;

select * from resume_from_apply_jobpost(*some id_jobpost*);
```

RA

$$\pi_{id\_user, email, resume}(\sigma_{id\_jobpost='jobpostid'}(users \bowtie apply\_job\_post))$$

TRC

$$\{ t \mid \exists u \in users (t[id\_user] = u[id\_user] \wedge \\ t[email] = u[email] \wedge t[resume] = u[resume] \wedge \\ \exists aj \in apply\_job\_post (aj[id\_jobpost] = 'jobpostid' \wedge u[id\_user] = aj[id\_user])) \}$$

DRC

$$\{ \langle iu, em, res \rangle \mid \exists fn, ln, pa, ad, ci, st, con, q, s, py, dob, age, desg, h, act, abt, sk ( \\ \langle iu, fn, ln, em, pa, ad, ci, st, con, q, s, py, dob, age, desg, res, h, act, abt, sk \rangle \in users \wedge \\ \exists ia, ij, ic, stat ( \langle ia, ij, ic, iu, stat \rangle \in apply\_job\_post \wedge ij = 'jobpostid')) \}$$

## Retrieve qualified jobs greater than some salary

```
delimiter //
create procedure job_from_minimumsalary(input_id_user int(11),
input_minimumsalary varchar(255))
begin
select companyname, jobtitle, description, minimumsalary,
maximumsalary, experience
from users
where users.id_user=input_id_user
inner join job_post
on users.qualification=job_post.qualification and
job_post.maximumsalary>input_minimumsalary;
end //
delimiter ;

call job_from_minimumsalary(*some user_id*, *some minimumsalary*);
```

RA

$$\pi_{\text{jobtitle,description,minimumsalary,maximumsalary,experience}}(\sigma_{\text{id\_user='userid' \wedge maximumsalary > 'minimumsalary'}}(\text{users} \bowtie \text{job\_post}))$$

TRC

$$\{t | \exists u \in \text{users} (u[\text{id\_user}] = \text{'userid'} \wedge \exists j \in \text{job\_post} (t[\text{jobtitle}] = j[\text{jobtitle}] \wedge t[\text{description}] = j[\text{description}] \wedge t[\text{minimumsalary}] = j[\text{minimumsalary}] \wedge t[\text{maximumsalary}] = j[\text{maximumsalary}] \wedge t[\text{experience}] = j[\text{experience}] \wedge u[\text{qualification}] = j[\text{qualification}] \wedge j[\text{maximumsalary}] > \text{'minimumsalary'})))\}$$

DRC

$$\{ \langle jt, desc, mins, maxs, ex \rangle \mid \exists iu, fn, ln, em, pa, ad, ci, st, con, q, s, py, dob, age, desg, res, h, act, abt, sk ( \langle iu, fn, ln, em, pa, ad, ci, st, con, q, s, py, dob, age, desg, res, h, act, abt, sk \rangle \in \text{users} \wedge iu = \text{'userid'} \wedge \exists ij, ic, crd ( \langle ij, ic, jt, desc, mins, maxs, ex, crd \rangle \in \text{job\_post} \wedge maxs > \text{'minimumsalary'}) ) \}$$

## Retrieve users with applications sent out

```
SELECT id_user,firstname,lastname
FROM users
WHERE id_user IN (SELECT id_user FROM apply_job_post);
```

RA

$$\pi_{\text{id\_user,firstname,lastname}}(\text{users} \bowtie \text{apply\_job\_post})$$

TRC

$$\{t | \exists u \in \text{users} (t[\text{id\_user}] = u[\text{id\_user}] \wedge t[\text{firstname}] = u[\text{firstname}] \wedge t[\text{lastname}] = u[\text{lastname}] \wedge \exists aj \in \text{apply\_job\_post} (u[\text{id\_user}] = aj[\text{id\_user}]))\}$$

DRC

$$\{ \langle iu, fn, ln \rangle \mid \exists em, pa, ad, ci, st, con, q, s, py, dob, age, desg, res, h, act, abt, sk ( \langle iu, fn, ln, em, pa, ad, ci, st, con, q, s, py, dob, age, desg, res, h, act, abt, sk \rangle \in \text{users} \wedge \exists ia, ij, ic, stat ( \langle ia, ij, ic, iu, stat \rangle \in \text{apply\_job\_post}) ) \}$$

## Retrieve cities with jobs available

```
SELECT city
FROM company
WHERE company.id_company IN (SELECT id_company FROM job_post);
```

RA

$$\pi_{\text{city}}(\text{company} \bowtie \text{job\_post})$$

TRC

$$\{t | \exists c \in \text{company} (t[\text{city}] = c[\text{city}] \wedge \exists j \in \text{job\_post} (c[\text{id\_company}] = j[\text{id\_company}])))\}$$

DRC

$$\{ \langle ci \rangle \mid \exists ic, n, cn, c, st, co, w, e, p, a, l, cd, ac ( \\ \langle ic, n, cn, c, st, ci, co, w, e, p, a, l, cd, ac \rangle \in \text{company} \wedge \\ \exists ij, jt, desc, mins, maxs, ex, crd ( \langle ij, ic, jt, desc, mins, maxs, ex, crd \rangle \in \text{job\_post} )) \}$$

## Check if company email exists

```
SELECT email
FROM company
WHERE email = '$email';
```

RA

$$\pi_{\text{email}}(\sigma_{\text{email} = '$email'}(\text{company}))$$

TRC

$$\{t | \exists c \in \text{company} (t[\text{email}] = c[\text{email}] \wedge c[\text{email}] = '$email')\}$$

DRC

$$\{ \langle e \rangle \mid \exists ic, n, cn, c, st, ci, co, w, p, a, l, cd, ac ( \langle ic, n, cn, c, st, ci, co, w, e, p, a, l, cd, ac \rangle \in \text{company} \wedge e = '$email') \}$$

## Check if user email exists

```
SELECT email
FROM users
WHERE email = '$email';
```

RA

$$\pi_{\text{email}}(\sigma_{\text{email} = '$email'}(\text{users}))$$

TRC

$$\{t | \exists u \in \text{users} (t[\text{email}] = u[\text{email}] \wedge u[\text{email}] = '$email')\}$$

DRC

$$\{ \langle em \rangle \mid \exists iu, fn, ln, pa, ad, ci, st, con, q, s, py, dob, age, desg, res, h, act, abt, sk ( \\ \langle iu, fn, ln, em, pa, ad, ci, st, con, q, s, py, dob, age, desg, res, h, act, abt, sk \rangle \in \text{users} \\ \wedge em = 'email') \}$$

## Retrieve jobpost details from matching.jobpost



```
SELECT *
FROM job_post
WHERE id_jobpost = '$_GET[id]';
```

RA

$$\sigma_{id\_jobpost=\$_GET[id]}(job\_post)$$

TRC

$$\{t|\exists j \in job\_post(t[id\_jobpost]=j[id\_jobpost] \wedge t[id\_company]=j[id\_company] \wedge t[jobtitle]=j[jobtitle] \wedge t[description]=j[description] \wedge t[minimumsalary]=j[minimumsalary] \wedge t[maximumsalary]=j[maximum salary] \wedge t[experience]=j[experience] \wedge t[qualification]=j[qualification] \wedge t[createdat]=j[createdat] \wedge j[id\_jobpost]=\$_GET[id])\}$$

DRC

$$\{ \langle ij, ic, jt, desc, mins, max, ex, q, crd \rangle \mid \langle ij, ic, jt, desc, mins, max, ex, q, crd \rangle \in job\_post \wedge ij = \$_GET[id] \}$$

### Retrieve company details

```
SELECT id_company, companyname, email, active
FROM company
WHERE email = '$email' AND password = '$password';
```

RA

$$\pi_{id\_company, companyname, email, active}(\sigma_{email=\$email \wedge password=\$password}(company))$$

TRC

$$\{t|\exists c \in company(t[id\_company]=c[id\_company] \wedge t[companyname]=c[companyname] \wedge t[email]=c[email] \wedge t[active]=c[active] \wedge c[email]=\$email)\}$$

DRC

$$\{ \langle ic, cn, e, ac \rangle \mid \exists n, c, st, ci, co, w, p, a, l, cd ( \langle ic, n, cn, c, st, ci, co, w, e, p, a, l, cd, ac \rangle \in company \wedge p = \$password \wedge e = \$email) \}$$

### Retrieve applications that user has applied for

```
SELECT *
FROM apply_job_post
WHERE id_user = '$_SESSION[id_user]' AND
id_jobpost = '$row[id_jobpost]';
```

RA

$$\sigma_{id\_user=\$_SESSION[id\_user] \wedge id\_jobpost=\$row[id\_jobpost]}(apply\_job\_post)$$

TRC

$$\{t|\exists aj \in apply\_job\_post(t[id\_apply]=aj[id\_apply] \wedge t[id\_jobpost]=aj[id\_jobpost] \wedge t[id\_company]=aj[id\_company] \wedge t[id\_user]=c[id\_user] \wedge t[status]=aj[status] \wedge aj[id\_user]=\$_SESSION[id\_user] \wedge id\_jobpost=\$row[id\_jobpost])\}$$

DRC

$$\{ \langle ia, ij, ic, iu, stat \rangle \mid \langle ia, ij, ic, iu, stat \rangle \in \text{apply\_job\_post} \wedge \\ iu = \text{'$_SESSION[id\_user]'} \wedge ij = \text{'$row[id\_jobpost]'} \}$$

### Retrieve cities in a state

```
SELECT *  
FROM cities  
WHERE state_id = "$_POST[id]";
```

RA

$$\sigma_{\text{state\_id} = \text{'$_POST[id]'}}(\text{cities})$$

TRC

$$\{ t \mid \exists ci \in \text{cities} (t[id] = ci[id] \wedge t[name] = ci[name] \wedge \\ t[state\_id] = ci[state\_id] \wedge c[state\_id] = \text{'$_POST[id]'}) \}$$

DRC

$$\{ \langle id, n, si \rangle \mid \langle id, n, si \rangle \in \text{cities} \wedge si = \text{'$_POST[id]'} \}$$

### Retrieve certain jobpost and their company details

```
SELECT * job_post  
INNER JOIN company  
ON job_post  
WHERE id_jobpost = '$_GET[id]';
```

RA

$$\sigma_{\text{id\_jobpost} = \text{'$_GET[id]'}}(\text{job\_post} \bowtie \text{company})$$

TRC

$$\{ t \mid \exists j \in \text{job\_post} (t[id\_jobpost] = j[id\_jobpost] \wedge t[id\_company] = j[id\_company] \wedge \\ t[jobtitle] = j[jobtitle] \wedge t[description] = j[description] \wedge \\ t[minimumsalary] = j[minimumsalary] \wedge t[maximumsalary] = j[maximumsalary] \wedge \\ t[experience] = j[experience] \wedge t[qualification] = j[qualification] \wedge \\ t[createdat] = j[createdat] \wedge j[id\_jobpost] = \text{'$_GET[id]'} \wedge \\ \exists c \in \text{company} (t[id\_company] = c[id\_company] \wedge \\ t[name] = c[name] \wedge t[companyname] = c[companyname] \wedge \\ t[country] = c[country] \wedge t[state] = c[state] \wedge \\ t[city] = c[city] \wedge t[contactno] = c[contactno] \wedge \\ t[website] = c[website] \wedge t[email] = c[email] \wedge \\ t[password] = c[password] \wedge t[aboutme] = c[aboutme] \wedge \\ t[logo] = c[logo] \wedge t[createdat] = c[createdat] \wedge t[active] = c[active])) \}$$

DRC

$$\{ \langle ij, ic, jt, desc, mins, max, ex, q, crd, n, cn, c, st, ci, co, w, e, p, a, l, cd, ac \rangle \mid \\ \langle ij, ic, jt, desc, mins, max, ex, q, crd \rangle \in \text{job\_post} \wedge \\ \langle ic, n, cn, c, st, ci, co, w, e, p, a, l, cd, ac \rangle \in \text{company} \wedge ij = \text{'$_GET[id]'} \}$$

### Triggers

The following are the triggers used for this project. Note that: Since the `id` uses auto incremented, we do not need to explicitly specify an ID, so we write `null` instead.

Add inserted to the logs when a new user is created.

```
CREATE TRIGGER 'insertLog'  
AFTER INSERT ON 'users'  
FOR EACH ROW  
INSERT INTO logs VALUES(null, NEW.id_user, "Inserted" , NOW())
```

Add updated to the logs when a user's detail(s) are updated

```
CREATE TRIGGER 'updateLog'  
AFTER UPDATE ON 'users'  
FOR EACH ROW  
INSERT INTO logs VALUES(null, NEW.id_user, "Updated" , NOW())
```

Add deleted to the logs when a user is deleted

```
CREATE TRIGGER 'deleteLog'  
AFTER DELETE ON 'users'  
FOR EACH ROW  
INSERT INTO logs VALUES(null, OLD.id_user, "Deleted" , NOW())
```

## Usage

The following are the steps to run this project yourself

1. Clone the [Github Repo](#)
2. Import the `git_repos_jobportal.sql` in the xampp mysql database
3. Paste the `Job_Portal` folder into `htdocs` folder. (found in WAMPP, XAMMP, MAMPP installation folder)
4. Start the Apache and Mysql services from the MAMPP server (mac), XAMPP server (windows).
5. Run the following commands in the terminal

```
cd Job-Portal
open db.php
```

6. Check that credentials for username and password are matching according to your system.
7. Go to `localhost` and job posts will now be visible.
8. Login with any of the below default accounts

| Type      | Username                | Password |
|-----------|-------------------------|----------|
| Admin     | admin                   | 12345    |
| Candidate | <a href="#">m@m.com</a> | m        |
| Company   | <a href="#">m@m.com</a> | m        |

## References

- [1] “Need for DBMS,” GeeksforGeeks, Aug. 23, 2016. <https://www.geeksforgeeks.org/need-for-dbms/> (accessed May 02, 2022).
- [2] “Schema Integration in DBMS,” GeeksforGeeks, Oct. 11, 2018. <https://www.geeksforgeeks.org/schema-integration-in-dbms/> (accessed May 01, 2022).
- [3] “Normal Forms in DBMS,” GeeksforGeeks, Jul. 08, 2015. <https://www.geeksforgeeks.org/normal-forms-in-dbms/> (accessed May 01, 2022).
- [4] F. Nelson, Job-Portal. 2021. Accessed: Apr. 08, 2022. [Online]. Available: <https://github.com/fuifilen/job-portal>
- [5] “Introduction of ER Model,” GeeksforGeeks, Oct. 13, 2015. <https://www.geeksforgeeks.org/introduction-of-er-model/> (accessed Apr. 28, 2022).
- [6] “Relational Model in DBMS,” GeeksforGeeks, Oct. 27, 2015. <https://www.geeksforgeeks.org/relational-model-in-dbms/> (accessed Apr. 21, 2022).
- [7] “Introduction of Relational Algebra in DBMS,” GeeksforGeeks, Jul. 02, 2015. <https://www.geeksforgeeks.org/introduction-of-relational-algebra-in-dbms/> (accessed Apr. 21, 2022).
- [8] “Functional Dependency and Attribute Closure,” GeeksforGeeks, Dec. 23, 2016. <https://www.geeksforgeeks.org/functional-dependency-and-attribute-closure/> (accessed Apr. 22, 2022).
- [9] “ACID Properties in DBMS,” GeeksforGeeks, Aug. 07, 2016. <https://www.geeksforgeeks.org/acid-properties-in-dbms/> (accessed Apr. 25, 2022).
- [10] “Concurrency Control in DBMS,” GeeksforGeeks, Dec. 29, 2015. <https://www.geeksforgeeks.org/concurrency-control-in-dbms/> (accessed Apr. 26, 2022).
- [11] “Database Management System (DBMS),” GeeksforGeeks. <https://www.geeksforgeeks.org/dbms/> (accessed Apr. 22, 2022).