



Bilkent University

CS353

DATABASE SYSTEMS

Project Design Report

**Company Interview and Employment Review Platform
Database System**

GROUP 5

Mehmet Sanisoğlu

Mehmet Selim Özcan

Ayça Begüm Taşçioğlu

Erdem Ege Maraşlı

Table of Contents

1.	Revised E/R Model	4
2.	Relational Schemas	6
2.1.	User	6
2.2.	Employee	7
2.3.	Employer	8
2.4.	Company	9
2.5.	Follows	10
2.6.	Works	11
2.7.	Job	12
2.8.	Project	13
2.9.	Photo	14
2.10.	Award	15
2.11.	Applies	16
2.12.	Review	17
2.13.	Responses	18
2.14.	Publishes	19
2.15.	Related	20
2.16.	Admin	21
2.17.	Salary_Review	22
2.18.	Benefits_Review	23
2.19.	General_Review	24
2.20.	Interview_Review	25
2.21.	Requests	26
2.22.	Member	27
3.	Functional Dependencies and Normalization of Tables	28
4.	Functional Components	28
4.1.	Use Cases/Scenarios	28
4.2.	Algorithms	30
4.3.	Data Structures	31
4.4.	Use-Case Diagram	31
5.	User Interface Design and Corresponding SQL Statements	32
5.1.	Login Page	32
5.2.	Register Page	33
5.3.	Reset Password Page	35
5.4.	Standard User Profile	36
5.5.	Company Profile Page	38
5.6.	Create Review Page	40
5.7.	Review Page	42
5.8.	Create Job Page	43

5.9.	Job Page	44
5.10.	Create Project Page	45
5.11.	Project Page	47
5.12.	Apply a Job Page	48
5.13.	Respond a Review Page	50
5.14.	Request to Remove a Review Page	52
5.15.	Remove a Review Page	53
5.16.	Applied Job List Page	54
5.17.	Posted Job List Page	55
5.18.	Searched Job List Page	57
5.19.	Review List Page	58
5.20.	Job's Review Page	60
5.21.	Project List Page	61
5.22.	Followers Page	62
5.23.	Applicant Page	63
6.	Advanced Database Components	64
6.1.	Views	64
6.1.1.	ReviewList View	64
6.1.2.	Applications View	64
6.1.3.	ParticipatedProjects View	64
6.1.4.	JobsWithHighSalaries View	64
6.1.5.	CompaniesWithHighRatings View	65
6.1.6.	ApplicableJobs View	65
6.2	Triggers	65
6.3	Constaints	66
7.	Implementation Plan	66
8.	Website	66

1. Revised E/R Model

We revised our E/R model according to assistant's feedback.

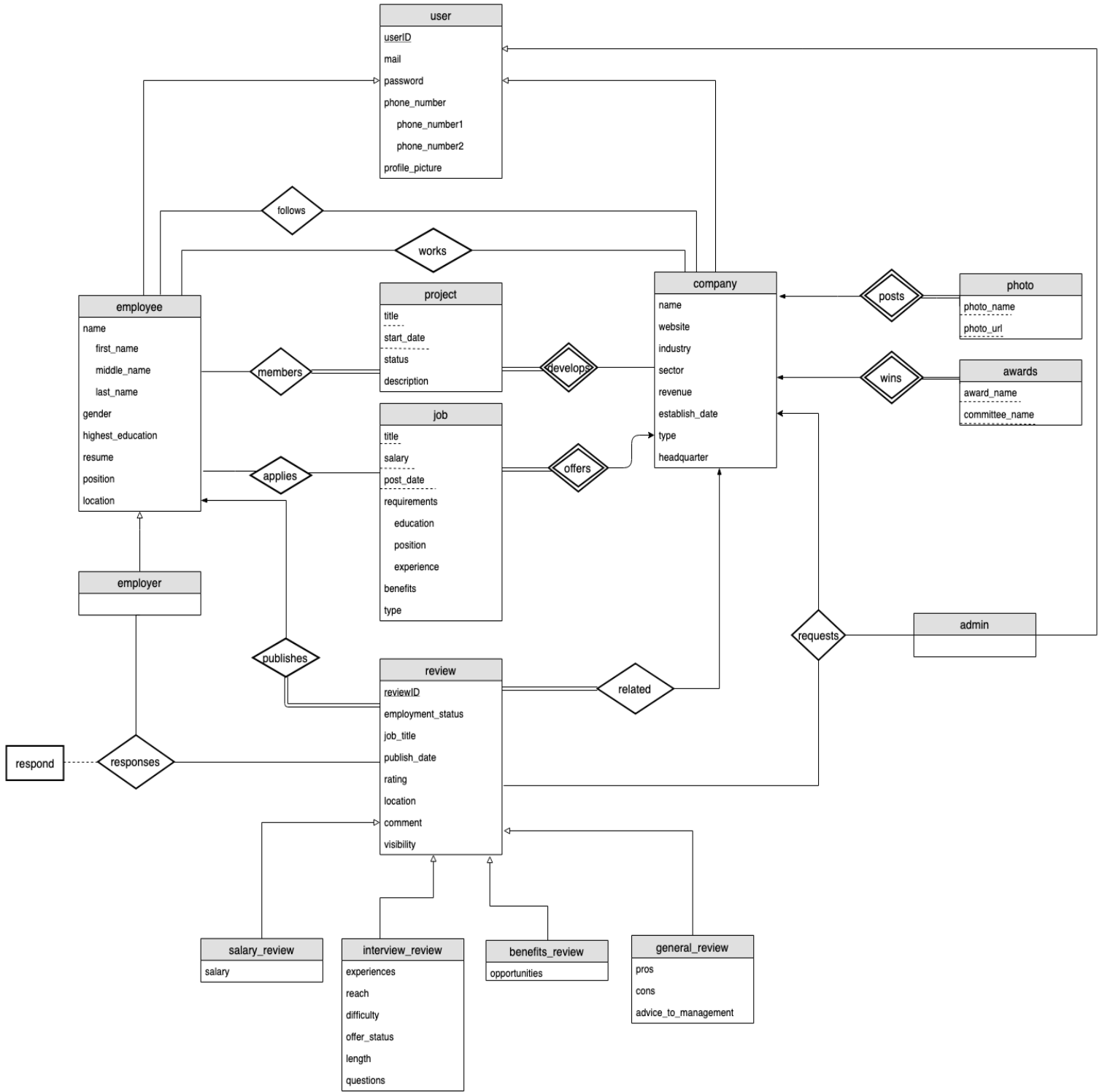
For E/R diagram:

1. Binary relationships between company-admin, company-review were converted to the ternary relationship.
2. total_followers() and size() method in company were deleted. The follows relationship is added between company and employee.
3. The member relationship between project and employee was converted to the total participation.
4. The publish relationship between review and employee was converted to the total participation.
5. The develop relationship between company and project was changed.
6. The related relationship between the company and the review was converted to the total participation.
7. The distance method was deleted from the job.
8. The admin became an instance of user. All the attributes in admin was deleted.
9. respond attribute was added to responses relationship.
10. photo entity's attributes were changed: "photo_name" , "photo_url" were added; "tag" was deleted.
11. award entity's attributes were changed: "award_name" , "committee_name" were added; "type" was deleted.

For Limitations:

We wrote down all limitations again as follows:

1. The Company Review System cannot be used without login.
2. System supports at most 50 000 company user.
3. System supports at most 200 000 standard user.
4. System supports at most 1000 reviews for single company.
5. Employer can only respond reviews about its own company.
6. Users can only have one profile picture.
7. Standard Users can upload only one resume.
8. Employees can work for at most 2 companies at the same time.
9. Standard User cannot apply a job if experience level of user is not sufficient for job.
10. Reviews' rating and difficulty level are between 0-10.



2. Relational Schemas

2.1 User

Relational Model:

user(userID, mail, password, phone_number1, phone_number2, profile_picture)

Functional Dependencies:

userID → mail, password, phone_number1, phone_number2, profile_picture

mail → password, phone_number1, phone_number2, profile_picture

Candidate Keys:

{ (userID), (mail) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE user(  
    userID      varchar(20) PRIMARY KEY,  
    mail        varchar(40) NOT NULL,  
    password    varchar(20) NOT NULL,  
    phone_number1  varchar(20),  
    phone_number2  varchar(20),  
    profile_picture  varchar(200) ) Engine=InnoDB;
```

2.2 Employee

Relational Model:

employee(employeeID, first_name, middle_name, last_name, gender, highest_education, resume, position, location)

Functional Dependencies:

employeeID -> first_name, middle_name, last_name, gender, highest_education, resume, position, location

Candidate Keys:

{{employeeID}}

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE employee(  
    employeeID      varchar(20) PRIMARY KEY,  
    first_name      varchar(40) NOT NULL,  
    middle_name     varchar(40),  
    last_name       varchar(40) NOT NULL,  
    gender          varchar(20),  
    highest_education varchar(40),  
    resume          varchar(40) NOT NULL,  
    position        varchar(40),  
    Location        varchar(40),  
    FOREIGN KEY(employeeID) REFERENCES user(userID) ) Engine=InnoDB;
```

2.3 Employer

Relational Model:

employer(employerID)

Functional Dependencies:

None

Candidate Keys:

{{employerID}}

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE employer(  
  employerID          varchar(20) PRIMARY KEY,  
  FOREIGN KEY(employerID) REFERENCES employee(employeeID) ) Engine=InnoDB;
```


2.4 Company

Relational Model:

company(companyID, name, website, industry, sector, revenue, establish_date, type, headquarter)

Functional Dependencies:

companyID -> name, website, industry, sector, revenue, establish_date, type, headquarter

Candidate Keys:

{ (companyID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE company(  
    companyID    varchar(20) PRIMARY KEY,  
    name         varchar(20) NOT NULL,  
    website      varchar(50),  
    industry     varchar(10) NOT NULL,  
    sector       varchar(10) NOT NULL,  
    revenue      double,  
    establish_date NOT NULL,  
    type         varchar(10) NOT NULL,  
    headquarter  varchar(10) NOT NULL,  
    FOREIGN KEY(companyID) REFERENCES user(userID) ) Engine=InnoDB;
```

2.5 Follows

Relational Model:

follows(employeeID, companyID)

Functional Dependencies:

None

Candidate Keys:

{ (employeeID, companyID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE follows(  
    employeeID      varchar(20),  
    companyID       varchar(20),  
    PRIMARY KEY(employeeID, companyID),  
    FOREIGN KEY(employeeID) REFERENCES employee(employeeID),  
    FOREIGN KEY(companyID) REFERENCES company(companyID) ) Engine=InnoDB;
```

2.6 Works

Relational Model:

works(employeeID, companyID)

Functional Dependencies:

None

Candidate Keys:

{ (employeeID, companyID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE works(  
    employeeID      varchar(20),  
    companyID       varchar(20),  
    PRIMARY KEY(employeeID, companyID),  
    FOREIGN KEY(employeeID) REFERENCES employee(employeeID),  
    FOREIGN KEY(companyID) REFERENCES company(companyID) ) Engine=InnoDB;
```

2.7 Job

Relational Model:

job(companyID, title, salary, post_date, education, position, experience, benefits, type)

Functional Dependencies:

(companyID, title, salary, post_date) -> education, position, experience, benefits, type

Candidate Keys:

{(companyID, title, salary, post_date)}

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE job(  
    companyID      varchar(20),  
    title          varchar(40),  
    salary         double,  
    post_date      date,  
    education      varchar(40),  
    position       varchar(20) NOT NULL,  
    experience     varchar(40),  
    benefits       varchar(40),  
    type          varchar(40) NOT NULL,  
    PRIMARY KEY(companyID, title, salary, post_date),  
    FOREIGN KEY(companyID) REFERENCES company(companyID) ) Engine=InnoDB;
```

2.8 Project

Relational Model:

project(companyID, title, start_date, status, description)

Functional Dependencies:

(companyID, title, start_date) -> status, description

Candidate Keys:

{(companyID, title, start_date)}

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE project(  
    companyID      varchar(20),  
    title          varchar(20),  
    start_date     date,  
    status         varchar(40),  
    description    varchar(40),  
    PRIMARY KEY(companyID, title, start_date),  
    FOREIGN KEY(companyID) REFERENCES company(companyID) ) Engine=InnoDB;
```

2.9 Photo

Relational Model:

photo(companyID, photo_name, photo_url)

Functional Dependencies:

None

Candidate Keys:

{ (companyID, photo_name, photo_url) }

Normal Form:

BCNF

Table Definition:

```
create table photo(  
    companyID    varchar(20),  
    photo_name   varchar(20),  
    photo_url    varchar(200),  
    PRIMARY KEY(companyID, photo_name, photo_url),  
    FOREIGN KEY(companyID) REFERENCES company(companyID) ) Engine=InnoDB;
```

2.10 Award

Relational Model:

award(companyID, award_name, committee_name)

Functional Dependencies:

None

Candidate Keys:

{{companyID, award_name, committee_name}}

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE award(  
    companyID          varchar(20),  
    award_name         varchar(20),  
    committee_name     varchar(20),  
    PRIMARY KEY(companyID, award_name, committee_name),  
    FOREIGN KEY(companyID) REFERENCES company(companyID) ) Engine=InnoDB;
```

2.11 Applies

Relational Model:

applies(employeeID, companyID, title, salary, post_date)

Functional Dependencies:

None

Candidate Keys:

{ (employeeID,companyID,title,salary,post_date) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE applies(  
    employeeID  varchar(20),  
    companyID   varchar(20),  
    title       varchar(40),  
    salary      double,  
    post_date   date,  
    PRIMARY KEY ( employeeID, companyID, title, salary, post_date ),  
    FOREIGN KEY ( employeeID ) REFERENCES employee(employeeID),  
    FOREIGN KEY ( companyID, title, salary, post_date ) references job(companyID, title,  
salary, post_date) ) Engine=InnoDB;
```


2.12 Review

Relational Model:

review(reviewID, employment_status, job_title, date, rating, location, comment, visibility)

Functional Dependencies:

reviewID -> employment_status, job_title, date, rating, location, comment, visibility

Candidate Keys:

{ (reviewID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE review(  
    reviewID          int PRIMARY KEY,  
    Employment_status bit NOT NULL,  
    job_title         varchar(40) NOT NULL,  
    publish_date      date NOT NULL,  
    rating            double NOT NULL,  
    location          varchar(40) NOT NULL,  
    comment           varchar(500) NOT NULL,  
    visibility        bit NOT NULL ) ) Engine=InnoDB;
```

2.13 Responses

Relational Model:

responses(reviewID, employerID, respond)

Functional Dependencies:

None

Candidate Keys:

{ (reviewID, employerID, respond) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE responses(  
    respond      varchar(200),  
    reviewID     int,  
    employerID   varchar(20),  
    PRIMARY KEY (reviewID, employerID, respond),  
    FOREIGN KEY (reviewID) references review(reviewID),  
    FOREIGN KEY (employerID) references employer(employerID) ) Engine=InnoDB;
```

2.14 Publishes

Relational Model:

publishes(reviewID, employeeID)

Functional Dependencies:

None

Candidate Keys:

{ (reviewID, employeeID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE publishes(  
    reviewID          int,  
    employeeID        varchar(20),  
    PRIMARY KEY (reviewID, employeeID),  
    FOREIGN KEY (reviewID) references review(reviewID),  
    FOREIGN KEY (employeeID) references employee(employeeID) ) Engine=InnoDB;
```

2.15 Related

Relational Model:

related(reviewID, companyID)

Functional Dependencies:

None

Candidate Keys:

{ (reviewID, companyID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE related(  
    reviewID          int,  
    companyID         varchar(20),  
    PRIMARY KEY ( reviewID, companyID),  
    FOREIGN KEY ( reviewID) references review(reviewID),  
    FOREIGN KEY ( companyID) references company(companyID) ) Engine=InnoDB;
```

2.16 Admin

Relational Model:

admin(adminID)

Functional Dependencies:

None

Candidate Keys:

{ (adminID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE admin(  
    adminID          varchar(20) PRIMARY KEY,  
    FOREIGN KEY(adminID) REFERENCES user(userID) ) Engine=InnoDB;
```

2.17 Salary_Review

Relational Model:

salary_review(reviewID, salary)

Functional Dependencies:

reviewID -> salary

Candidate Keys:

{ (reviewID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE salary_review(  
    reviewID          int PRIMARY KEY,  
    salary             double NOT NULL,  
    FOREIGN KEY(reviewID) REFERENCES review(reviewID) ) Engine=InnoDB;
```

2.18 Benefits_Review

Relational Model:

benefits_review(reviewID, opportunities)

Functional Dependencies:

reviewID -> opportunities

Candidate Keys:

{ (reviewID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE benefits_review(  
    reviewID          int PRIMARY KEY,  
    opportunities      varchar(100) NOT NULL,  
    FOREIGN KEY(reviewID) REFERENCES review(reviewID) ) Engine=InnoDB;
```

2.19 General_Review

Relational Model:

general_review(reviewID, pros, cons, advice_to_management)

Functional Dependencies:

reviewID -> pros, cons, advice_to_management

Candidate Keys:

{ (reviewID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE general_review(  
    reviewID                int PRIMARY KEY,  
    pros                    varchar(100) NOT NULL,  
    cons                    varchar(100) NOT NULL,  
    advice_to_management    varchar(200),  
    FOREIGN KEY(reviewID) REFERENCES review(reviewID) ) Engine=InnoDB;
```


2.20 Interview_Review

Relational Model:

interview_review(reviewID, experiences, reach, difficulty, offer_status, length, questions)

Functional Dependencies:

reviewID -> experiences, reach, difficulty, offer_status, length, questions

Candidate Keys:

{ (reviewID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE interview_review(  
    reviewID          int PRIMARY KEY,  
    experiences        varchar(200) NOT NULL,  
    reach              varchar(20) NOT NULL,  
    difficulty         int NOT NULL,  
    offer_status       bit NOT NULL,  
    length             int NOT NULL,  
    Questions          varchar(200) NOT NULL,  
    FOREIGN KEY(reviewID) REFERENCES review(reviewID) ) Engine=InnoDB;
```

2.21 Requests

Relational Model:

requests(reviewID, companyID, adminID)

Functional Dependencies:

None

Candidate Keys:

{ (reviewID, companyID, adminID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE requests (  
    reviewID          int,  
    companyID         varchar(20),  
    adminID           varchar(20),  
    PRIMARY KEY (reviewID, companyID, adminID),  
    FOREIGN KEY (reviewID) references review(reviewID),  
    FOREIGN KEY (companyID) references company(companyID),  
    FOREIGN KEY (adminID) references admin(adminID) ) Engine=InnoDB;
```

2.22 Members

Relational Model:

members(employeeID, companyID, title, start_date)

Functional Dependencies:

None

Candidate Keys:

{{employeeID, companyID, title, start_date}}

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE members (  
    employeeID      varchar(20),  
    companyID       varchar(20),  
    title           varchar(40),  
    start_date      date,  
    PRIMARY KEY ( employeeID, companyID, title, start_date),  
    FOREIGN KEY ( employeeID ) references employee(employeeID)  
    FOREIGN KEY ( companyID ) references project(companyID) ) Engine=InnoDB;
```

3. Functional Dependencies and Normalization of Tables

All tables are in Boyce-Codd Normal Form, therefore, there is no need for normalization.

4. Functional Components

4.1 Use Cases / Scenarios

Company Interview and Employment Review Platform Database System, there are 5 types of actors which are User, Employee, Employer, Company and the Admin. The roles of each user type have both differences and similarities. In order to use the system, all types of actors should register and login to the system. The offered features of the system according to actor types are follows:

User

- A user can register and login to the system with userID and password.
- A user can reach his/her profile page which includes the personal information such as the mail address, userID, phone number, and profile picture.
- A user can change his/her password in profile page.
- An user can edit the account settings; s/he can change profile picture, phone number, password, or mail address.
- A user can search a job.

Employee

- An employee can register and login to the system with userID and password.
- An employee can reach his/her profile page which includes the personal information such as the mail address, userID, name, gender, highest education information, resume, position, location, phone number, and profile picture.
- An employee can change his/her password in profile page.
- An employee can edit the account settings; s/he can change profile picture, phone number, password, or mail address and s/he can upload resume.
- An employee can list the companies with company names, websites, industries, sectors, revenue, establish dates, types and headquarters.
- An employee can view a specific company.
- An employee can follow a specific company.
- An employee can apply for a job, in this case the employee should fill out the specifications which are posted by the company. This application requirements can be education level, the applied position, or experience.
- An employee can make a review. S/he should specify the review type such as salary review, interview review, benefits review or a general review. After

deciding the review type, s/he should fill out the review specifications which are related with the review type and rate the overall experience.

Employer

- An employer can register and login to the system with userID and password.
- An employer can reach his/her profile page which includes the personal information such as the mail address, userID, name, gender, highest education information, resume, position, location, phone number, profile picture.
- An employer can change his/her password in profile page.
- An employer can edit the account settings; s/he can change profile picture, phone number, password, or mail address and s/he can upload resume.
- An employer can list the companies with company names, websites, industries, sectors, revenue, establish dates, types and headquarters.
- An employer can view a specific company.
- An employer can follow a specific company.
- An employer can respond to the reviews which are related with his/her company.

Company

- A company can register and login to the system with userID and password.
- A company can reach his/her profile page which includes the account information such as the name, website, industry, sector, revenue, establish_date, type, headquarter.
- A company can change its password in profile page.
- A company can edit the account settings; it can change profile picture, phone number, password, or mail address.
- A company can view their developed projects.
- A company can list the past applications of an applicant to one if their offered jobs.
- A company can list all of their own reviews.
- A company can send a request to an admin for a specific review to be removed.
- A company can post a new job offer where they need to specify the required information about the job.
- A company can accept an applicant for an applied job.
- A company can reject an applicant for an applied job.
- A company can post a photo.
- A company can develop a new project where they need to specified the required project information.
- A company can view the list of their followers.

- A company can get a list view of the applicants of one of their jobs

Admin

- An admin can login to the system with userID and password.
- An admin can reach his/her profile page which includes the account information such as the name, website, industry, sector, revenue, establish_date, type, headquarter.
- An admin can edit the account settings; s/he can change profile picture, phone number, password, or mail address.
- An admin can get a list view of all the review removal request s/he has received.
- An admin can accept a removal request where afterwards the corresponding company is notified.
- An admin can reject a removal request where afterwards the corresponding company is notified.

4.2 Algorithms

Photos

Users can have profile picture and companies can publish pictures in our system. We plan to implement this feature via storing such images in our server's local file system. In our database, we store the image paths of these images. We access these mapped path information via SQL statement and load the image from the local file system of our server. We will use stored image paths to access it.

Resume

Employees must upload their resume in our system. We again plan to implement this feature via storing such files in our server's local file system. In our database, we store the file paths of these files. We access these mapped path information via SQL statement and load the file from the local file system of our server. We will use stored file paths to access it.

Sorting/Searching

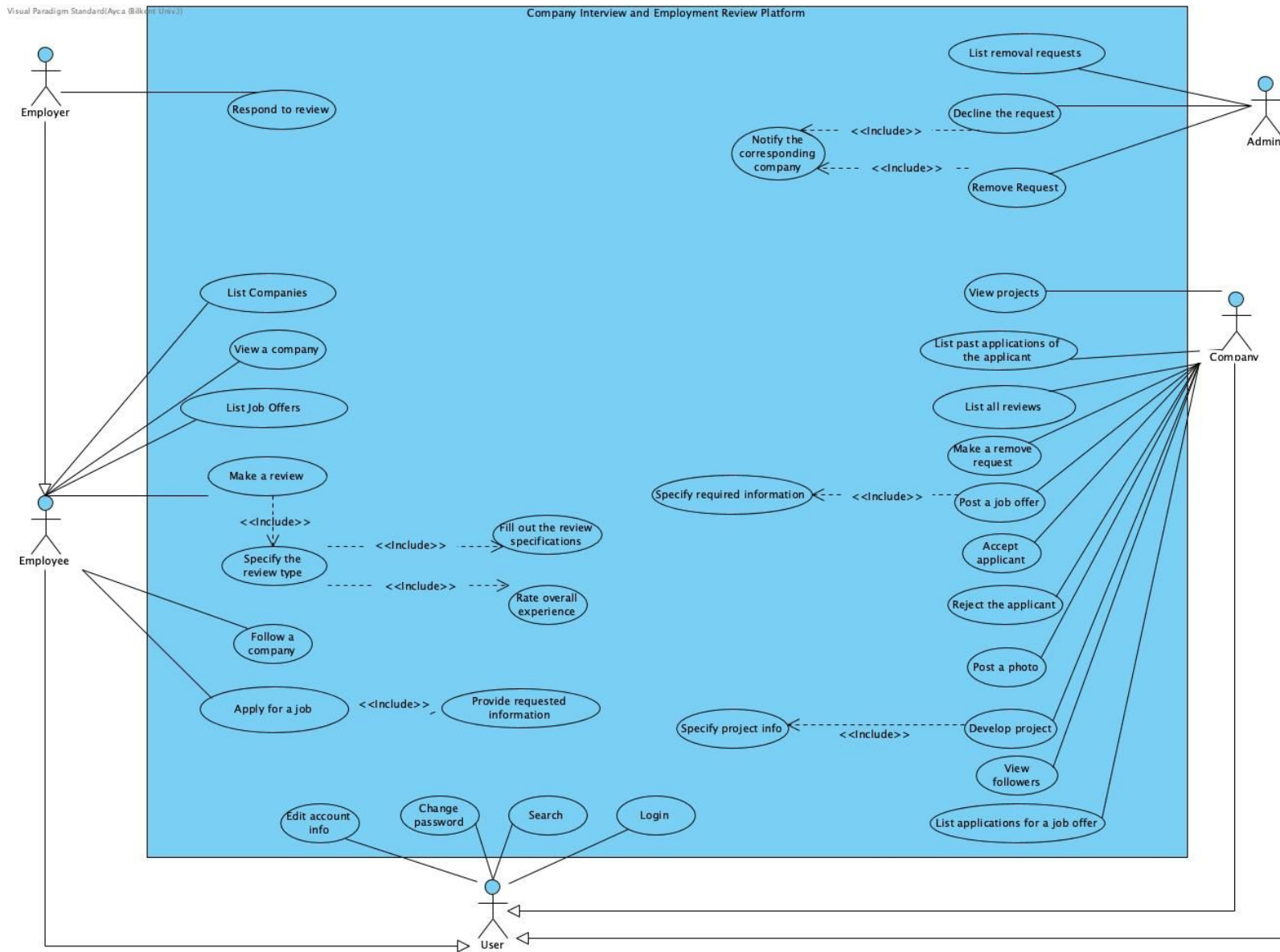
We are planning to write search and sort algorithms for user to use our system easily. They will just write their desired keyword and possible solutions will shown to the user.

Also, we are planning to write sort algorithms for users to sort jobs according to their desire, education and field. We are planning to write these algorithms with SQL statements.

4.3 Data Structures

We are using SQL's built-in data types such as NUMERIC, DATE, BIT, STRING.

4.4 Use-Case Diagram



5. User Interface Design and Corresponding SQL Statements

5.1 Login Page

Login

← → ↻ 🔍

LOGIN

UserID:

Password:

[Forget your password?](#)

Don't have an account? [Register now](#)

Accessible by: Anyone entered the website

Available actions: Login the system

Available navigations: Reset password, register

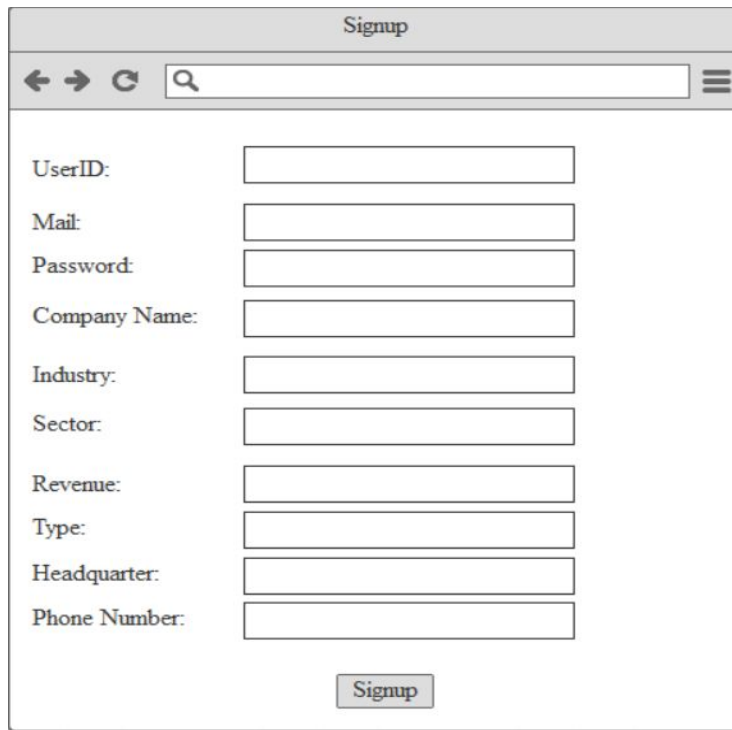
Procedure: User fills out the userID and password sections. If the provided information checks out, user is directed to home page. If not error message will pop up.

Inputs: @userID, @password

SQL Statements

```
SELECT * FROM user WHERE userID = @userID AND password = @password;
```


5.2 Register Page



A web browser window titled "Signup" with a navigation bar containing back, forward, and refresh buttons, a search bar, and a menu icon. The main content area contains a registration form with the following fields: UserID, Mail, Password, Company Name, Industry, Sector, Revenue, Type, Headquarter, and Phone Number. Each field is represented by a text input box. A "Signup" button is located at the bottom center of the form.

UserID:	<input type="text"/>
Mail:	<input type="text"/>
Password:	<input type="text"/>
Company Name:	<input type="text"/>
Industry:	<input type="text"/>
Sector:	<input type="text"/>
Revenue:	<input type="text"/>
Type:	<input type="text"/>
Headquarter:	<input type="text"/>
Phone Number:	<input type="text"/>

Figure 1: Standard user register page



A web browser window titled "Signup" with a navigation bar containing back, forward, and refresh buttons, a search bar, and a menu icon. The main content area contains a registration form with the following fields: UserID, Mail, Password, Company Name, Industry, Sector, Revenue, Type, Headquarter, and Phone Number. Each field is represented by a text input box. A "Signup" button is located at the bottom center of the form.

UserID:	<input type="text"/>
Mail:	<input type="text"/>
Password:	<input type="text"/>
Company Name:	<input type="text"/>
Industry:	<input type="text"/>
Sector:	<input type="text"/>
Revenue:	<input type="text"/>
Type:	<input type="text"/>
Headquarter:	<input type="text"/>
Phone Number:	<input type="text"/>

Figure 2 : Company Register Page

Accessible by : Anyone entered the website who had not logged in.

Available actions: Register the system.

Available navigations: Login

Procedure: First user is asked to declare the profile type : Standard user or Company. If user has selected standard user, s/he is prompted to choose a unique user id, valid password with a valid e-mail address and also their name, surname, gender, location, phone number. If user gets approval from system, s/he presses “Create Account” button and is directed to login page. If the system does not approve the provided informations such as not uniqueness or invalid e-mail address, there is an error message.

If user has selected the company type; unique userID, unique password, valid mail,name of the company, industry type, sector, revenue, type, headquarter, phone number are required.

Inputs: @userID, @password, @mail

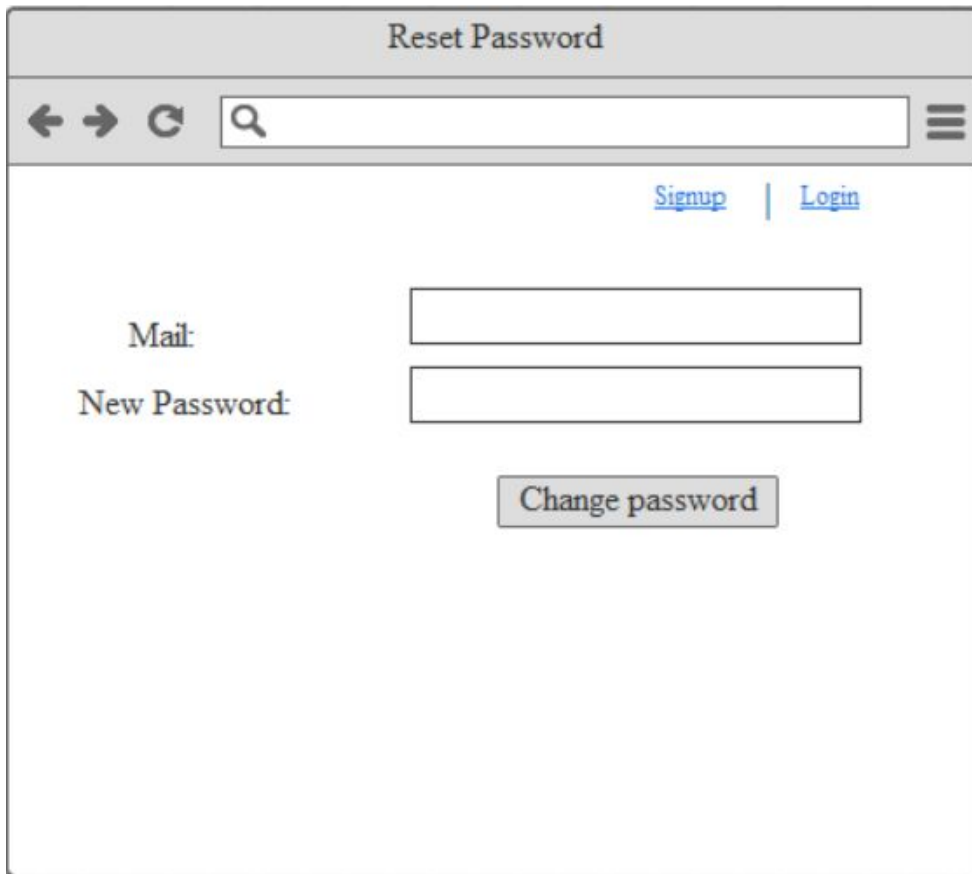
SQL Statements

```
INSERT INTO user(  
userID, mail, password, phone_number1, phone_number2 profile_picture)  
VALUES(@userID, @mail, @password, @phone_number1, NULL, NULL);
```

```
INSERT INTO employee(  
userID, first_name, middle_name, last_name, gender, highest_education, resume, position,  
location)  
VALUES(@userID, @first_name, @middle_name, @last_name, @gender, NULL, NULL, NULL,  
@location);
```

```
INSERT INTO company(  
userID, name, website, industry, sector, revenue, establish_date, type, headquarter)  
VALUES(@userID, @name, NULL, @sector, @revenue, NULL, @type, @headquarter);
```

5.3 Reset Password Page



The image shows a web browser window titled "Reset Password". The browser's address bar contains navigation icons (back, forward, refresh) and a search bar. In the top right corner of the page, there are links for "Signup" and "Login". The main content area features two input fields: the first is labeled "Mail:" and the second is labeled "New Password:". Below these fields is a button labeled "Change password".

Accessible by: Anyone entered the website

Available actions: Reset password

Available navigations: Login, Register

Procedure: If user wants to change password, an e-mail including the link to new password page is sent to user's provided mail address. If user clicks the link s/he can change the password.

Inputs: @mail, @new_password

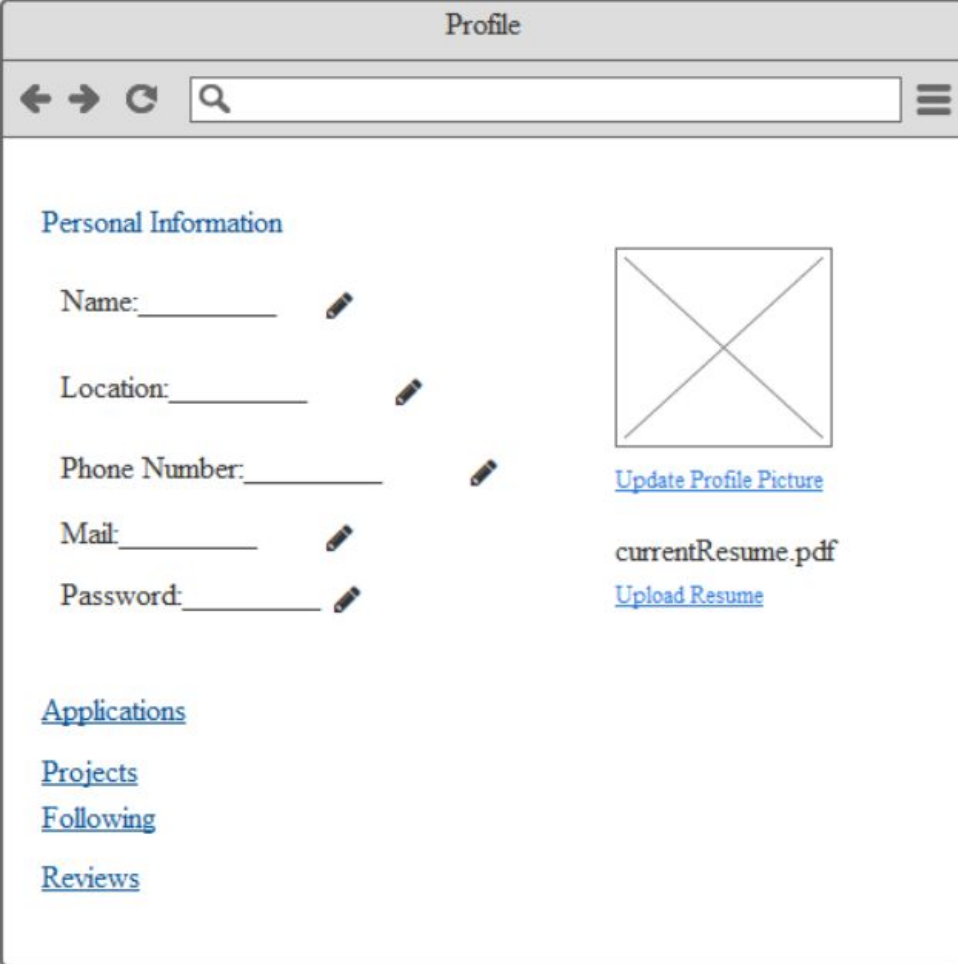
SQL Statements

UPDATE user

SET password = @new_password

WHERE mail = @mail;

5.4 Standard User Profile Page



The screenshot shows a web browser window titled "Profile". The browser's address bar contains a search icon and a menu icon. The main content area is divided into two sections. The top section, "Personal Information", contains five input fields: "Name:", "Location:", "Phone Number:", "Mail:", and "Password:". Each input field has a small pencil icon to its right, indicating it can be edited. To the right of these fields is a large square placeholder for a profile picture, marked with a large 'X'. Below the placeholder are two links: "Update Profile Picture" and "currentResume.pdf". Below the "currentResume.pdf" link is another link: "Upload Resume". The bottom section, "Applications", contains four links: "Projects", "Following", and "Reviews".

Accessible by: Standard User

Available actions: Update resume , Update profile picture

Available navigations: Job List, Company List, Project List, Review List

Procedure: User can access the list of applied job and project lists, followed company list and also reviews. Also user can change personal informations.

Inputs: @employeeID, @companyID, @new_resume, @new_picture

SQL Statements

SELECT *

FROM employee

WHERE employeeID = @employeeID;

INSERT INTO works(

companyID, employeeID)

VALUES(@companyID,@employeeID);

UPDATE employee

SET resume = @new_resume

WHERE employeeID = @employeeID;

UPDATE user

SET profile_picture = @new_picture

WHERE userID = @employeeID;

5.5 Company Profile Page

The screenshot shows a web browser window titled "Profile". The browser's address bar is empty, and the page content is organized into two main sections. The top section, titled "Personal Information", contains a list of form fields for company details: "Company Name:", "Website Link:", "HQ Location:", "Establish Date:", "Phone Number:", "Mail:", "Password:", "Sector:", "Industry:", "Revenue:", and "Type:". Each field is followed by a small edit icon (a pencil). To the right of these fields is a placeholder for a profile picture, represented by a square with a large 'X' inside, and a link below it that says "Update Profile Picture". The bottom section of the page contains a vertical list of links: "My Jobs", "My Employees", "My Projects", "Followers", and "Reviews".

Accessible by: Company User

Available actions: Update company information, display lists.

Available navigations: Job List, Project List, Review List, Follower List, Employee List

Procedure: Company user can access the list of applicants, reviews, followers, projects, jobs and employees of its own. Also company can see and update its information

Inputs: @companyID, @new_picture

SQL Statements

SELECT *

FROM company

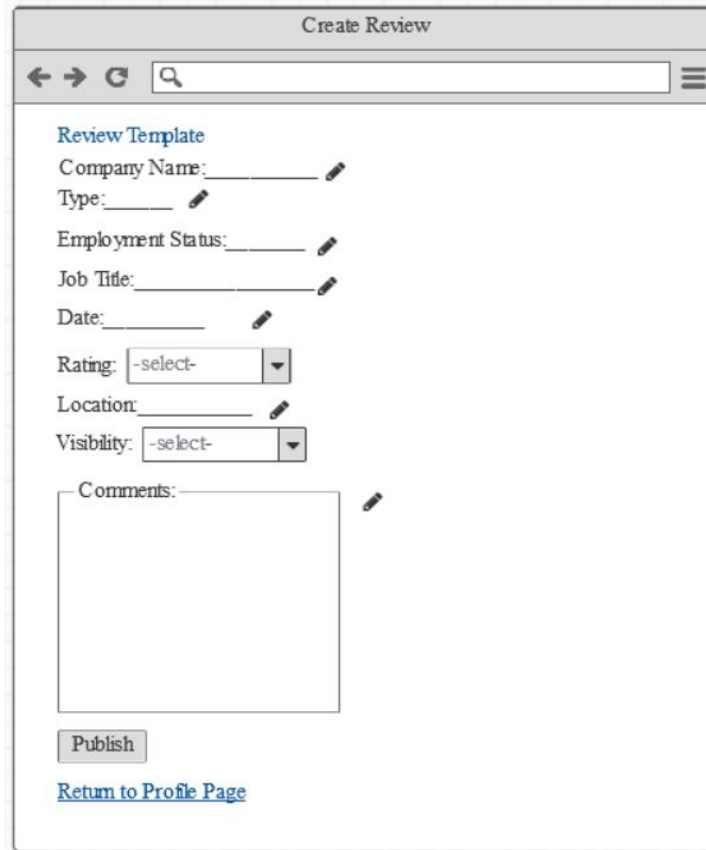
```
WHERE companyID = @companyID;
```

```
UPDATE user
```

```
SET profile_picture = @new_picture
```

```
WHERE userID = @companyID;
```

5.6 Create Review Page



The screenshot shows a web browser window titled "Create Review". The browser's address bar contains a search icon and a menu icon. The form itself has a title "Review Template" in blue. It contains several input fields, each with a small edit icon to its right: "Company Name:", "Type:", "Employment Status:", "Job Title:", "Date:", "Rating:" (a dropdown menu showing "-select-"), "Location:", and "Visibility:" (a dropdown menu showing "-select-"). Below these is a "Comments:" label followed by a large text area. At the bottom of the form is a "Publish" button and a blue link labeled "Return to Profile Page".

Accessible by: Standard User

Available actions: Create a new review with specified type.

Available navigations: Standard User Profile page

Procedure: Standard user creates a review about a company by specifying review type and filling review's attributes.

Inputs: @reviewID, @employment_status,@ job_title, @publish_date, @rating, @location, @comment, @visibility, @companyID, @employeeID

SQL Statements

```
INSERT INTO review(  
reviewID, employment_status, job_title, publish_date, rating, location, comment, visibility)  
VALUES(@reviewID, @employment_status,@ job_title, @publish_date, @rating, @location,  
@comment, @visibility);
```



```
INSERT INTO publishes(  
reviewID, employeeID)  
VALUES(@reviewID,@employeeID);
```

```
INSERT INTO related(  
reviewID, companyID)  
VALUES(@reviewID,@companyID);
```

5.7 Review Page

The screenshot shows a web browser window titled "Review Page". The browser's address bar contains a search icon and a menu icon. The page content includes a section titled "Review Information" with the following labels and input fields: "Company Name:", "Type:", "Employment Status:", "Job Title:", "Date:", "Rating:", "Location:", "Visibility:", and "Publisher Name:". Below these is a "Comments:" label followed by a large text area. At the bottom of the page is a link labeled "Return to Profile Page".

Accessible by: All users

Available actions: -

Available navigations: User profile page

Procedure: Shows a review of a designated type such as interview,salary,benefits or general about a company from a publisher user.

Inputs: @reviewID, @employeeID

SQL Statements

SELECT *

FROM publishes

WHERE employeeID = @employeeID AND reviewID =@reviewID

5.8 Create Job Page

Create Job

← → ↻ 🔍 ☰

Job Template

Company Name: _____ ✎

Title: _____ ✎

Salary: _____ ✎

Education: _____ ✎

Position: _____ ✎

Status: _____ ✎

Experience: _____ ✎

Benefits: _____ ✎

Type: _____ ✎

[Return to Profile Page](#)

Accessible by: Company User

Available actions: Create a new job

Available navigations: Company Profile page

Procedure: Company user posts new job offerings by specifying job's attributes.

Inputs: @companyID, @title, @salary, @post_date, @education, @position, @experience, @benefits, @type

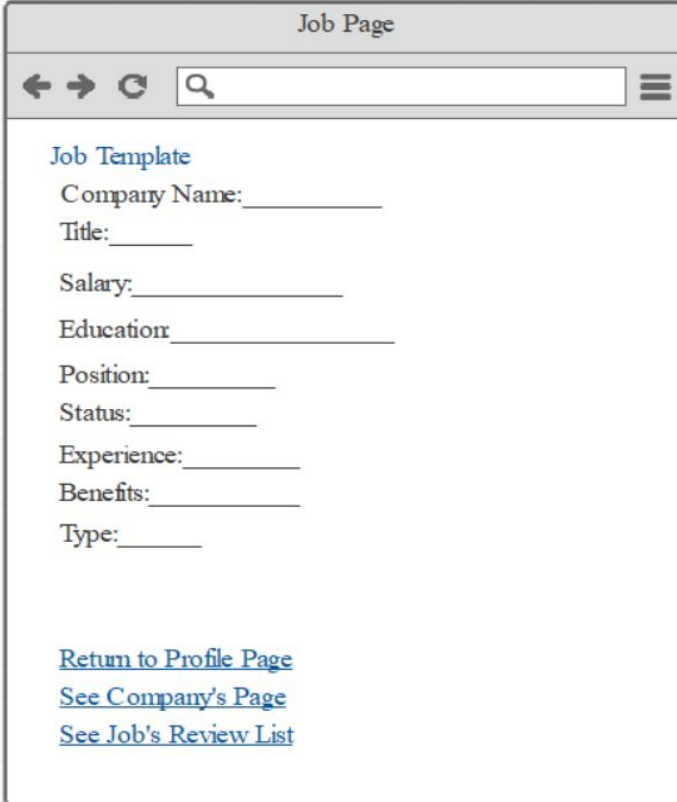
SQL Statements

```
INSERT INTO job(
```

```
companyID, title, salary, post_date, education, position, experience, benefits, type)
```

```
VALUES(@companyID, @title, @salary, @post_date, @education, @position, @experience,  
@benefits, @type);
```

5.9 Job Page



The screenshot shows a web browser window titled "Job Page". The browser's address bar contains navigation icons (back, forward, refresh) and a search bar. The main content area of the page is titled "Job Template" in blue text. Below this title, there is a form with several input fields, each preceded by a label: "Company Name:", "Title:", "Salary:", "Education:", "Position:", "Status:", "Experience:", "Benefits:", and "Type:". Each label is followed by a horizontal line representing an input field. At the bottom of the form, there are three blue hyperlinks: "Return to Profile Page", "See Company's Page", and "See Job's Review List".

Accessible by: All users

Available actions: -

Available navigations: User Profile Page, Company Page, Job's Review Page

Procedure: Shows a job description from a Company.

Inputs: @reviewID, @employeeID

SQL Statements

```
SELECT *
```

```
FROM job
```

```
WHERE companyID = @companyID AND title = @title AND salary = @salary AND post_date =  
@post_date;
```

5.10 Create Project Page

Project Template

Project Title: _____

Title: _____

Project Description: _____

Create

[Return to Profile Page](#)

Accessible by: Company User

Available actions: Create a new project

Available navigations: Company Profile page

Procedure: Company user develops new projects by specifying project's attributes and members.

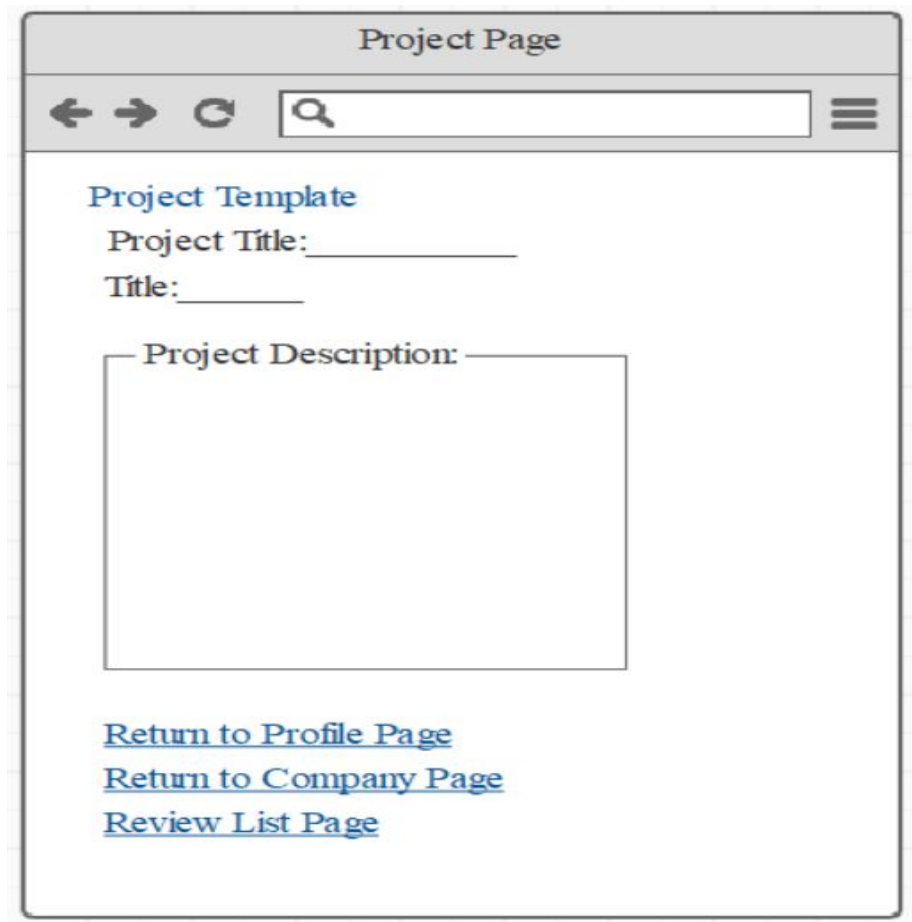
Inputs: @companyID, @title, @start_date, @status, @description

SQL Statements

```
INSERT INTO project(  
companyID, title, start_date, status, description)  
VALUES(@companyID, @title, @start_date, @status, @description);
```

```
INSERT INTO members(  
companyID, employeeID, title, start_date)  
VALUES(@companyID, @title, @start_date);
```

5.11 Project Page



The screenshot shows a web browser window titled "Project Page". The browser's address bar contains navigation icons (back, forward, refresh) and a search bar. The main content area of the page is titled "Project Template" in blue. Below this title, there are three input fields: "Project Title:" followed by a text input, "Title:" followed by a text input, and "Project Description:" followed by a large text area. At the bottom of the page, there are three blue underlined links: "Return to Profile Page", "Return to Company Page", and "Review List Page".

Accessible by: All users

Available actions: -

Available navigations: User Profile Page, Company Page

Procedure: Shows a project description from a Company.

Inputs: @companyID, @title, @start_date

SQL Statements

```
SELECT *
```

```
FROM project
```

```
WHERE companyID = @companyID AND title = @title AND start_date = @start_date
```

5.12 Apply a Job Page

Apply Job

← → ↻ 🔍 ☰

Job:
Company Name: Microsoft
Title: _____
Salary: _____
Education: _____
Position: _____

[Jobs](#)
[Company Page](#)
[Return to Profile](#)

Accessible by: Standard User

Available actions: Apply a job

Available navigations: Job Page, Company Profile Page, Standard User Profile Page

Procedure: Standard user sends application request.

Inputs: @employeeID, @companyID, @title, @salary, @post_date

SQL Statements

INSERT INTO applies(

employeeID, companyID, title, salary, post_date)

VALUES(@employeeID,@companyID, @title, @salary, @post_date);

SELECT *

FROM job

WHERE companyID = @companyID AND title = @title AND salary = @salary AND post_date = @post_date;

5.13 Respond a Review Page

The screenshot shows a web browser window titled "Respond Review". The browser's address bar contains a search icon and a menu icon. The form itself has a "Review Template" section with the following fields: "Company Name:", "Type:", "Employment Status:", "Job Title:", "Date:", "Rating:", "Location:", and "Visibility:". Below these is a "Comments:" label followed by a large text input area. Underneath the comments is a "Write Response:" label followed by another large text input area, which has a small pencil icon to its right. A "Submit" button is located below the response text area. At the bottom of the form, there are four links: "Reviews", "Return to Profile", "Return to Company Page", and "See Publisher's Profile".

Accessible by: Employer User

Available actions: Respond a review

Available navigations: Review Page, Company Page, Writer's Profile Page, Employer's own Page

Procedure: Employer can access the review that belongs to its own company. S/he can respond it by filling the provided text area.

Inputs: @employerID, @reviewID, @respond

SQL Statements

```
INSERT INTO responses(  
  employerID, reviewID, respond)  
VALUES(@employerID, @reviewID, @respond);
```

5.14 Request to Remove a Review Page

The screenshot shows a web browser window titled "Request to Remove a Review". The browser's address bar is empty, and the page content is as follows:

Review Information

Company Name: _____

Type: _____

Employment Status: _____

Job Title: _____

Date: _____

Rating: _____

Location: _____

Visibility: _____

Publisher Name: _____

Comments: _____

[Reviews](#)

[Return to Profile](#)

[Return to Company Page](#)

[See Publisher's Profile](#)

Accessible by: Company User

Available actions: Send request to admin regarding removal of review

Available navigations: Review Page, Company Profile Page, Writer's Profile Page,

Procedure: Company fills out a removal request form about its own company and sends it to admin.

Inputs: @reviewID, @companyID, @adminID

SQL Statements

```
INSERT INTO requests(  
reviewID, companyID, adminID)  
VALUES(@reviewID, @companyID, @adminID);
```

5.15 Remove a Review Page

The screenshot shows a web browser window titled "Remove Review Page". The browser's address bar contains navigation icons (back, forward, refresh) and a search icon. The main content area displays the text "Review Title: _____" followed by a blue hyperlink "A Review Link" and a grey "Delete" button. Below this, there are five more blue hyperlinks: "Return to Admin Page", "Reviews", "Return to Profile", "Return to Company Page", and "See Publisher's Profile".

Accessible by: Admin User

Available actions: Deletes a review

Available navigations: Review Page, Company Profile Page, Writer's Profile Page, Admin Profile Page

Procedure: Admin can remove a review that requested by company itself

Inputs: @reviewID

SQL Statements

DELETE FROM review

WHERE reviewID = @reviewID;

5.16 Applied Job List Page

Applied Jobs List Page

Applied Jobs List

Job Title: _____	Company Name: _____	Link to Job Page	Delete
Job Title: _____	Company Name: _____	Link to Job Page	Delete
Job Title: _____	Company Name: _____	Link to Job Page	Delete
Job Title: _____	Company Name: _____	Link to Job Page	Delete
Job Title: _____	Company Name: _____	Link to Job Page	Delete
Job Title: _____	Company Name: _____	Link to Job Page	Delete

[Reviews](#)

[Return to Profile](#)

Accessible by: Standard User

Available actions: Update the applications

Available navigations: Job Page, Standard User Profile Page, Review Page

Procedure: User can see every application that s/he has been made also can cancel the application

Inputs: @employeeID, @companyID, @title, @salary, @post_date

SQL Statements

```
SELECT *;
```

```
FROM applies
```

```
WHERE employeeID = @employeeID;
```

```
DELETE FROM applies
```

```
WHERE userID = @userID AND companyID = @companyID AND title = @title AND salary =  
@salary AND post_date = @post_date
```

5.17 Posted Job List Page

The screenshot shows a web browser window titled "Posted Job List Page". The browser's address bar contains navigation icons (back, forward, refresh) and a search icon. Below the browser window, the page content is displayed. It starts with a heading "Posted Jobs List" in blue. Below this heading is a table with six rows. Each row contains three columns: "Job Title:" followed by a text input field, "Company Name:" followed by a text input field, and a "Link to Job Page" (a blue hyperlink) and a "Delete" button (a grey button). Below the table, there are three blue hyperlinks: "Applicant List", "Reviews", and "Return to Profile".

Job Title:	Company Name:	Link to Job Page	Delete
<input type="text"/>	<input type="text"/>	Link to Job Page	<button>Delete</button>
<input type="text"/>	<input type="text"/>	Link to Job Page	<button>Delete</button>
<input type="text"/>	<input type="text"/>	Link to Job Page	<button>Delete</button>
<input type="text"/>	<input type="text"/>	Link to Job Page	<button>Delete</button>
<input type="text"/>	<input type="text"/>	Link to Job Page	<button>Delete</button>
<input type="text"/>	<input type="text"/>	Link to Job Page	<button>Delete</button>

[Applicant List](#)
[Reviews](#)
[Return to Profile](#)

Accessible by: Company User

Available actions: Update the job offerings

Available navigations: Job Page, Company Profile Page, Applicant List Page, Review Page

Procedure: Company can see every jobs that it has been offered also can change its applicability

Inputs: @companyID, @title, @salary, @post_date

SQL Statements

```
SELECT *;
```

```
FROM job
```

```
WHERE companyID = @companyID;
```

```
DELETE FROM job
```

```
WHERE companyID= @companyID AND title = @title AND salary = @salary AND post_date =  
@post_date
```


5.18 Searched Job List Page

The screenshot shows a web browser window titled "Searched Job List Page". At the top, there is a search bar with a magnifying glass icon and a placeholder text "Q keyword". Below the search bar, the text "Results for 'keyword'" is displayed. The main content area contains a table with six rows of job listings. Each row has four columns: "Job Title:", "Company Name", "Location:", and "Link to Job Page". To the right of the "Link to Job Page" column, there are two buttons: "Apply" for the first row and "Delete" for the remaining five rows. Below the table, there are two links: "Reviews" and "Return to Profile".

Job Title:	Company Name	Location:	Link to Job Page	Action
Job Title: _____	Company Name	Location: _____	Link to Job Page	Apply
Job Title: _____	Company Name	Location: _____	Link to Job Page	Delete
Job Title: _____	Company Name	Location: _____	Link to Job Page	Delete
Job Title: _____	Company Name	Location: _____	Link to Job Page	Delete
Job Title: _____	Company Name	Location: _____	Link to Job Page	Delete
Job Title: _____	Company Name	Location: _____	Link to Job Page	Delete

[Reviews](#)
[Return to Profile](#)

Accessible by: Every User

Available actions: See the searched job offerings

Available navigations: Job Page, Publisher Company Page, User's own Profile Page, Review Page

Procedure: User searches certain jobs with the search bar. Results comes with this page. It shows job's important attributes such as company name and location.

Inputs: @companyID, @title, @salary, @post_date

SQL Statements

```
SELECT name, location;  
  
FROM job INNER JOIN company  
  
WHERE title = @title;
```

5.19 Review List Page

Review List Page

← → ↻ 🔍

☰

User Name

Rating: __	Company Name: _____	Display	Delete
Rating: __	Company Name: _____	Display	Delete
Rating: __	Company Name: _____	Display	Delete
Rating: __	Company Name: _____	Display	Delete
Rating: __	Company Name: _____	Display	Delete
Rating: __	Company Name: _____	Display	Delete
Rating: __	Company Name: _____	Display	Delete
Rating: __	Company Name: _____	Display	Delete
Rating: __	Company Name: _____	Display	Delete

[See Company Page](#)

[Return to Profile](#)

Accessible by: Standard User

Available actions: Update the review

Available navigations: Job Page, Company Page, Standard User Profile Page

Procedure: User can see every review that s/he has been made also can update the review

Inputs: @userID, @reviewID

SQL Statements

```
SELECT *;
```

FROM publishes

WHERE userID = @userID;

DELETE FROM publishes

WHERE userID= @userID AND reviewID = @reviewID;

5.20 Job's Review Page

The screenshot shows a web browser window titled "Job Review Page". The browser's address bar contains navigation icons (back, forward, refresh) and a search icon. Below the browser window, the page content is displayed. At the top, there is a label "Job Name". Below this, there is a table with 8 rows. Each row contains three elements: a "Rating:" followed by a text input field, a "Publisher's Name:" followed by a text input field, and a "Display" button. At the bottom of the page, there are three hyperlinks: "Return to Profile", "See Job Page", and "See Company Page".

Rating: <input type="text"/>	Publisher's Name: <input type="text"/>	Display
Rating: <input type="text"/>	Publisher's Name: <input type="text"/>	Display
Rating: <input type="text"/>	Publisher's Name: <input type="text"/>	Display
Rating: <input type="text"/>	Publisher's Name: <input type="text"/>	Display
Rating: <input type="text"/>	Publisher's Name: <input type="text"/>	Display
Rating: <input type="text"/>	Publisher's Name: <input type="text"/>	Display
Rating: <input type="text"/>	Publisher's Name: <input type="text"/>	Display
Rating: <input type="text"/>	Publisher's Name: <input type="text"/>	Display

[Return to Profile](#)
[See Job Page](#)
[See Company Page](#)

Accessible by: Every User

Available actions: -

Available navigations: Job Page, Company Page, Standard User Profile Page

Procedure: User can see every reviews that have been made about a certain job

Inputs: @companyID, @reviewID

SQL Statements

```
SELECT *;
```

```
FROM review R, job J
```

```
WHERE reviewID = @reviewID AND R.job_title = J.title AND J.companyID = @companyID;
```

5.21 Project List Page

The screenshot shows a web browser window titled "Project List Page". The browser's address bar contains a search icon and a search input field. Below the browser window, the page content is displayed. At the top, there is a label "Company Name" in blue. Below this, there is a table with six rows. Each row contains four columns: "Project Title:", "Start Date:", "Status:", and a blue hyperlink "Link to Project Page". The table is enclosed in a rectangular border. Below the table, there is a blue hyperlink "Return to Profile".

Project Title:	Start Date:	Status:	Link to Project Page
Project Title:_____	Start Date:_____	Status:_____	Link to Project Page
Project Title:_____	Start Date:_____	Status:_____	Link to Project Page
Project Title:_____	Start Date:_____	Status:_____	Link to Project Page
Project Title:_____	Start Date:_____	Status:_____	Link to Project Page
Project Title:_____	Start Date:_____	Status:_____	Link to Project Page
Project Title:_____	Start Date:_____	Status:_____	Link to Project Page

[Return to Profile](#)

Accessible by: Company User

Available actions: Update the project offerings

Available navigations: Company Profile Page, ProjectPage

Procedure: Company can see every projects which it develops

Inputs: @companyID, @title, @start_date

SQL Statements

SELECT *

FROM members

WHERE .companyID = @companyID AND title = @title AND start_date = @start_date

DELETE FROM project

WHERE companyID= @companyID AND title = @title AND start_date = @start_date;

5.22 Followers Page

Followers Page

← → ↻ 🔍 ☰

Company Name

User Name: _____	Job Title: _____	Link to Profile
User Name: _____	Job Title: _____	Link to Profile
User Name: _____	Job Title: _____	Link to Profile
User Name: _____	Job Title: _____	Link to Profile
User Name: _____	Job Title: _____	Link to Profile
User Name: _____	Job Title: _____	Link to Profile

[Return to Company Profile](#)

Accessible by: Company User

Available actions: -

Available navigations: Company Profile Page, User Profile Page

Procedure: Company can see its followers.

Inputs: @companyID

SQL Statements

```
SELECT E.name, E.employeeID
FROM follows AS F, employee E
WHERE F.companyID = @companyID AND
F.employeeID = E.employeeID;
```

5.23 Applicant Page

The screenshot shows a web browser window titled "Applicant Page". The browser's address bar contains navigation icons (back, forward, refresh) and a search icon. Below the browser window, the page content is displayed. At the top, the text "Job Title" is shown in blue. Below this, there is a table with six rows. Each row contains a "User Name:" label followed by a text input field and a blue hyperlink labeled "Link to Profile". At the bottom of the page, there is a blue hyperlink labeled "Return to Company Profile".

Accessible by: Company User

Available actions: Accept applicant, Reject Applicant,

Available navigations: Company Profile Page, User Profile Page

Procedure: Company can see applicants to its job offers.

Inputs: @companyID, @title, @salary, @post_date

SQL Statements

```
SELECT *
```

```
FROM employee AS E
```

```
WHERE E.employeeID IN( SELECT A.employeeID
```

```
FROM applies AS A
```

```
WHERE A.employeeID = E.employeeID and A.companyID =@companyID
```

6. Advanced Database Components

6.1 Views

6.1.1 ReviewsList View

ReviewsList view for standard user to see his/her former reviews.

```
CREATE VIEW ReviewsList AS(  
    SELECT R.job_title, R.date, R.rating, R.comment  
    FROM publishes AS P, review as R, employee AS E  
    WHERE P.reviewID = R.reviewID AND  
    E.employeeID = P.employeeID  
);
```

6.1.2 Applications View

Application view for employee to view his/her former applications.

```
CREATE VIEW Applications AS(  
    SELECT C.name, A.title  
    FROM applies AS A, employee AS E, company as C  
    WHERE A.employeeID = E.employeeID AND  
    A.companyID = C.companyID  
);
```

6.1.3 ParticipatedProjects View

ParticipatedProjects view for employee to view his/her projects.

```
CREATE VIEW ParticipatedProjects AS(  
    SELECT M.title  
    FROM members AS M, employee AS E  
    WHERE E.employeeID = M.employeeID  
);
```

6.1.4 JobsWithHighSalaries View

This view will be used for listing the all jobs whose salaries are higher than the average.

```
CREATE VIEW JobsWithHighSalaries AS(  

```



```

SELECT title
FROM job
WHERE salary > (AVG(salary) FROM job)
);

```

6.1.5 CompaniesWithHighRatings View

CompaniesWithHighRatings View to list all companies whose ratings are higher than the average

```

CREATE VIEW CompaniesWithHighRatings AS(
    SELECT C.name
    FROM company as C, review AS R, related as R1
    WHERE C.companyID = R1.companyID AND
    R.reviewID = R1.reviewID AND
    R.rating > (AVG(rating) FROM review)
);

```

6.1.6 ApplicableJobs View

ApplicableJobs View to show applicable jobs to employee. These jobs will be customized with the employee's education level.

```

CREATE VIEW ApplicableJobs AS(
    SELECT *
    FROM job AS J
    WHERE J.education in ( SELECT E.education
                           FROM employee AS E
                           WHERE E.education = J.education)
);

```

6.2 Triggers

- When a Review is added, updated or deleted, the relations (publishes, related and responses) will be updated.
- When an application is rejected or accepted, the corresponding application is removed from "applies" relation.
- When a company accept the job application, user is inserted to "works" relation with corresponding company.
- When a project is added, updated or deleted by company, "members" relationship will be updated.

6.3 Constraints

11. The Company Review System cannot be used without login.
12. System supports at most 50 000 company user.
13. System supports at most 200 000 standard user.
14. System supports at most 1000 reviews for single company.
15. Employer can only respond reviews about its own company.
16. Users can only have one profile picture.
17. Standard Users can upload only one resume.
18. Employees can work for at most 2 companies at the same time.
19. Standard User cannot apply a job if experience level of user is not sufficient for job.
20. Reviews' rating and difficulty level are between 0-10.

7. Implementation Plan



At data layer we will use MySQL server in our project as database management system. For application logic and user interface we will code in PHP and a small amount of JavaScript.

8. Website

<https://github.com/aeyc/Company-Interview-and-Employment-Review-Platform-Database-System>