VIETNAMESE GERMAN UNIVERISTY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEER

IT Jobs Website Application Report

Student 1: Nguyen Dai Minh - 10421037
Student 2: Nguyen Khoi Nguyen - 10421097
Student 3: Pham Nguyen Quy Nam - 10421093
Student 4: Pham Tuan Kiet - 10421033

December 13, 2024

Content

1 Introduction **4**

2 Requirements Gathering **6**

2.1 Use Case Diagram 6
2.2 Use Case Analysis 7
2.3 Sequence Diagram 19
2.3.1 Search Jobs 19
2.3.2 Register as Employer 20
2.3.3 Register as Jobseeker 21
2.3.4 Login to the website 22
2.3.5 Edit profile 23
2.3.6 Search Jobs for registered user 24
2.3.7 View jobs application details 25
2.3.8 Job application 25
2.3.9 Withdraw job application 26
2.3.10 Post jobs 26
2.3.11 Delete jobs 27
2.3.12 View jobs 27
2.3.13 View applicants info 28
2.4 Software Architecture 28
2.5 Deployment Diagram 29
2.6 Entity-Relationship Diagram 29
2.7 User Interface Mock-up 30
2.7.1 Main Page 30
2.7.2 Login Page 31
2.7.3 Jobseeker Registration Page 32
2.7.4 Employer Registration Page 33
2.7.5 Employer Page 34
2.7.6 Jobseeker Page 34

3 Back-end Programming **35**

3.1 Login 35
3.2 Register 36
3.2.1 Register for Employer 36
3.2.2 Register as Jobseeker 37
3.3 Manage applicants 39
3.4 Manage jobs 41
3.5 Manage applications 43
3.6 Manage employers 44

4 Front-end Programming 47
4.1 Main Page 47
4.2 Login Page 48
4.3 Jobseeker Registration Page 49
4.4 Employer Registration Page 50
4.5 Jobseeker Profile 51
4.6 Employer Profile 54

5 Back-end Testing 57
5.1 Search Jobs 57
5.1.1 Test case 1: Keyword search successful 57
5.1.2 Test case 2: Keyword search return null 58
5.2 Register as Employer 58
5.2.1 Test case 1: Register Successful 58
5.2.2 Test case 2: Register fail with missing information 59
5.3 Register as Jobseeker 60
5.3.1 Test case 1: Register Successful 60
5.3.2 Test case 2: Register fail with missing information 61
5.4 Edit profile 61
5.4.1 Test case 1: Edit Profile Jobseeker 61
5.4.2 Test case 2: Edit Profile Employer 62
5.5 Job application 62
5.5.1 Test case 1: Job application is already in 62
5.5.2 Test case 2: Job application is new 63
5.6 Post jobs 64
5.6.1 Test case 1: Post Jobs successful 64
5.6.2 Test case 2: Post Jobs unsuccessful 64
5.7 Delete jobs 64
5.7.1 Test case 1: Delete successful 64
5.7.2 Test case 2: Delete fail 65

6 Front-end Testing 66
6.1 Main Page 66
6.2 Login Page 66
6.3 Jobseeker Registration Page 66
6.4 Employer Registration Page 67
6.5 Employer Page 68
6.6 Jobseeker Page 69

7 Security 70

8 Change Management 72

8.1 Updated Use Case Diagram 73
8.2 Updated Use Case Analysis 74
8.3 Updated Sequence Diagram 75
8.3.1 Updated ER Diagram 76
8.4 Updated UI 76
8.4.1 Updated Employer Page 76
8.4.2 Updated Jobseeker Page 77
8.5 Updated Notification Page 77
8.6 Revision History 78

1 Introduction

Application: Online IT job seeking application

The journey from graduation to securing one's first job in the IT industry can be fraught with obstacles. Several factors contribute to the high rate of unemployment among IT freshers, including:

1. Decreasing Demand for Entry-Level Positions: In an increasingly competitive job market, companies may prioritize hiring candidates with prior experience or specialized skills, leaving entry-level positions in high demand but low supply.

2. Focus on Senior and Junior Positions: While there may be ample opportunities for senior and junior positions within the IT industry, fresh graduates often struggle to compete with candidates who possess more experience or advanced qualifications.

3. Difficulty in Establishing Contact with Employers: Many IT freshers face challenges in effectively presenting themselves to potential employers, whether due to lack of networking opportunities, inadequate resume or interview preparation, or limited access to job postings.

ITJ Ltd recognizes the pressing need to address these challenges and empower both job seekers and employers in the IT industry. Through its comprehensive job seeking services, ITJ Ltd offers innovative solutions to facilitate meaningful connections and streamline the recruitment process.

Employers partnering with ITJ Ltd can expect a range of benefits, including:

- Job Posting Services: Employers can post detailed job listings, including information about the position, required qualifications, and desired skills, ensuring that job seekers have access to relevant and up-to-date job opportunities.

- Applicant Management Tools: ITJ Ltd provides employers with robust applicant management tools, allowing them to view applicant profiles, track application statuses, and make informed hiring decisions. Employers can easily accept or reject applicants, schedule interviews, and communicate with candidates directly through the platform.

- Profile Management: Employers have the flexibility to update their profiles, showcasing their company culture, values, and job offerings. This allows job seekers to gain insights into potential employers and make informed decisions about their applications.

Job seekers utilizing ITJ Ltd's services can expect comprehensive support throughout their job search journey, including:

- Job Search and Application: Job seekers have access to a vast database of job listings, searchable by keywords, location, and industry. They can apply to jobs directly through the platform, streamlining the application process and increasing their visibility to potential employers.

- Profile Enhancement: Job seekers can create personalized profiles, highlighting their skills, education, and experience. They have the option to upload resumes, cover letters, and other relevant documents, allowing them to present a comprehensive overview of their qualifications to potential employers.

- Job Details and Updates: ITJ Ltd provides job seekers with detailed job descriptions,

including information about job responsibilities, qualifications, and application deadlines. Job seekers receive notifications about new job postings and updates, ensuring they stay informed about relevant opportunities. By bridging the gap between job seekers and employers, ITJ Ltd's job seeking services have the potential to transform the IT job market and address the challenges faced by IT freshers. Some of the key benefits and impacts of these services include:

1. Increased Access to Job Opportunities: IT freshers gain access to a wider range of job opportunities, including entry-level positions and internships, through ITJ Ltd's platform. This increased accessibility enhances their chances of securing meaningful employment opportunities within the IT industry.

2. Streamlined Recruitment Process: Employers benefit from a streamlined recruitment process, facilitated by ITJ Ltd's applicant management tools. By providing a centralized platform for posting jobs, reviewing applications, and communicating with candidates, employers can save time and resources while ensuring they attract and hire the best talent for their organizations.

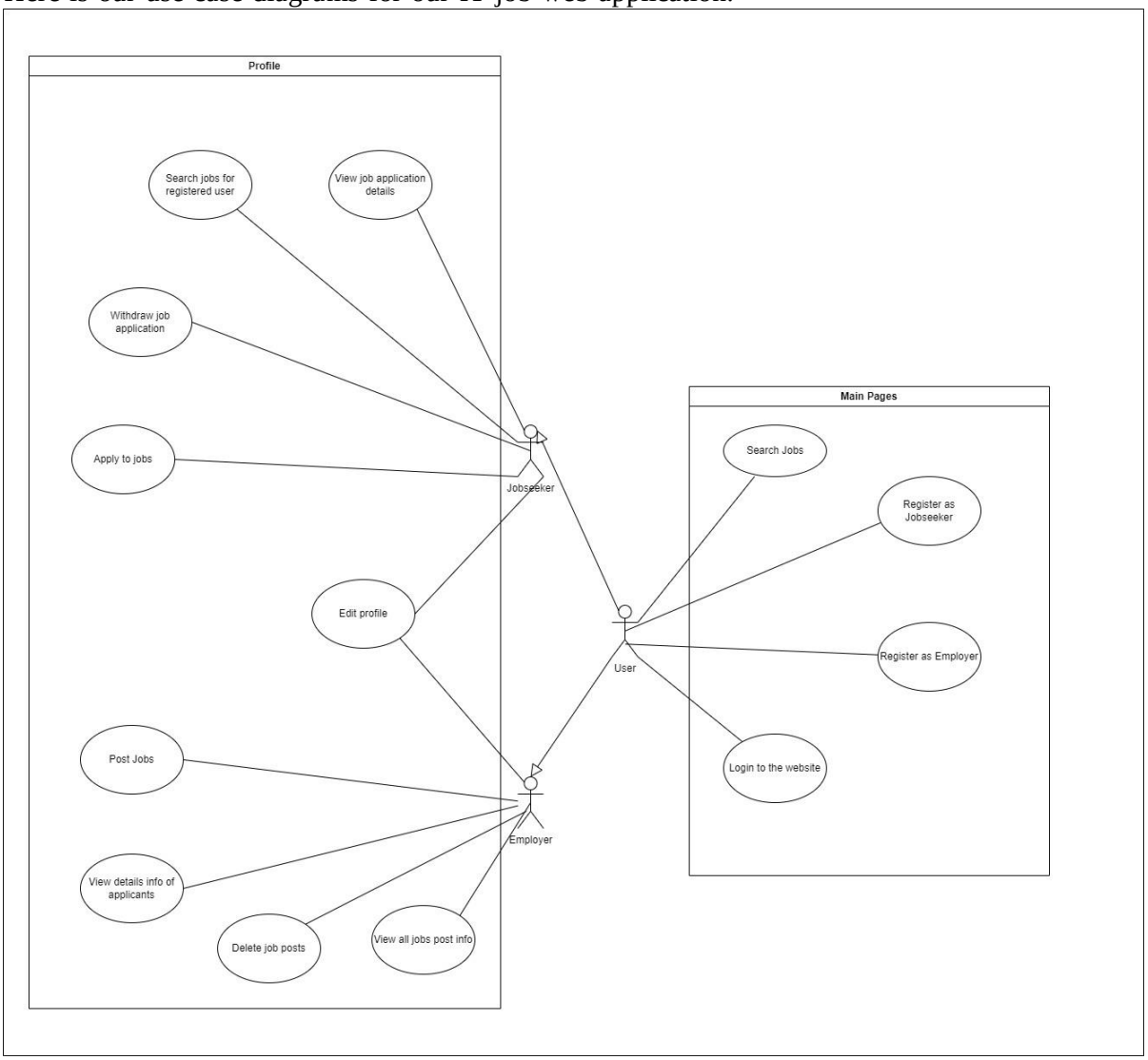
3. Improved Candidate-Employer Matching: ITJ Ltd's platform utilizes advanced algorithms and personalized profiles to match candidates with relevant job opportunities. This improved matching process increases the likelihood of successful hires and fosters long-term relationships between employers and employees.

4. Enhanced Professional Development: Job seekers utilizing ITJ Ltd's services gain access to resources and support to enhance their professional development. From resume writing tips to interview preparation workshops, ITJ Ltd empowers job seekers to present themselves effectively to potential employers and advance their careers within the IT industry.

2 Requirements Gathering

2.1 Use Case Diagram

Here is our use case diagrams for our IT job web application:



2.2 Use Case Analysis

Name	Search jobs
Description	This use case is for user to type a keyword into the search bar to find the jobs that match their search words.
Data Return	List of strings
Trigger	Users must enter a search term and press Search
Pre-conditions	Users must be on the main page of the web application
Post-conditions	A list of jobs matching the search terms will appear on the screen or none in case there are no jobs matching the search terms.
Basic Flow	1. The user enter the keyword in the search bar 2. The application database will compare the keyword with jobs.title or employer.ename and output result to the front end 3. The web app will output a list of jobs matching the user search terms 4. User can click on login to login or create an account
Alternative Flow	1. At step 3, if there are no jobs matching, the front end will display sorry, there are no jobs matching the search terms

Name	Register as Employer
Description	This use case is for user to register as an employer.
Data Return	None
Trigger	Users must click on become an Employer
Pre-conditions	Users must be on the employer registration page of the web application
Post-conditions	User will successfully register as an employer and transfer back to the main page of the application.
Basic Flow	1. The user navigate to the employer registration page of the web application 2. A form containing company information, login information, contact person, industry, location as well as brief description about the company will appear for the user to fill in and choose (employer.e_name, employer.executive, employer.address, employer.Industry, employer.phone, employer.location) 3. User can click on register to finish the registration 4. The database will receive information and store it in the employer table 5. Upon successful registration, the user will be redirect to the main page
Alternative Flow	1. At step 2, if the information entered is not in the correct format users will be warned.

Name	Register as Jobseeker
Description	This use case is for user to register as a job seeker.
Data Return	None
Trigger	Users must click on become a Jobseeker
Pre-conditions	Users must be on the job seeker registration page of the web application
Post-conditions	User will successfully register as a job seeker and transfer back to the main page of the application.
Basic Flow	1. The user navigate to the jobseeker registration page of the web application 2. A form containing personal information such as jobseeker.log_id, jobseeker.name, jobseeker.phone, jobseeker.location, jobseeker.master_edu, jobseeker.skills, jobseeker.basic_edu, jobseeker.experience 3. User can click on register to finish the registration 4. The database will receive information and store it in the jobseeker table 5. Upon successful registration, the user will be redirect to the main page
Alternative Flow	1. At step 2, if the information entered is not in the correct format users will be warned.

Name	Login to the website
Description	This use case is for user to login to the web-site.
Data Return	None
Trigger	Users must click on login
Pre-conditions	Users must be on the login page of the web application
Post-conditions	User will successfully login and redirect to account profile.
Basic Flow	1. The user navigate to the login page of the web application 2. User will enter their login credentials (login.email, login.password) 3. The database will check if the credentials enter is correct 4. User will be redirected to the account profile of either jobseeker or employer
Alternative Flow	1. At step 3, if the credentials entered is not correct, usesrs will have to enter the credentials again

Name	Edit profile
Description	This use case involves updating the basic information of both employers and jobseekers in the database.
Data Return	None
Trigger	Users must click on "Edit profile" in the menu task bar.
Pre-conditions	Users must register as either employers or jobseekers to updating the information in the database
Post-conditions	Employers or jobseekers will successfully update their account information after clicking on the "Confirm" button.
Basic Flow	1. The employer or jobseeker click on the "update" button on the profile UI. 2. The employer or jobseeker will able to see the option which can be modified modify in the profile (employer.e_name, employer.industry, employer.address, employer.executive, employer.phone, employer.e_type, jobseeker.log_id, jobseeker.name, jobseeker.phone, jobseeker.location, jobseeker.master_edu, jobseeker.skills, jobseeker.basic_edu, jobseeker.experience). 3. The employer or jobseeker can click confirm to finish edit profile.

Name	Search job for registered user
Description	This use case describes the scenario when the jobseeker looks for a job.
Data Return	List of objects containing strings
Trigger	User must type the job name on the search bar and hit on "Search" button or user can click on "Advanced search"
Pre-conditions	Users must register as jobseeker and is logged in.
Post-conditions	None
Basic Flow	1. The user searches for a specific job based on what they type into the search bar. 2. The app retrieves the jobs based on the jobseeker's input (employer.e_name, employer.industry, employer.e_type) 3. User will be presented with a list of jobs matching the input
Alternative Flow	On step 2, user can use advance search. 1. The user searches for a specific job based on industries, skills needed, etc. (employer.e_name, employer.industry, employer.e_type, job.title, job.Industry). 2. The app filters out jobs that match job-seeker's desire.

Name	View job application details
Description	This use case describes the scenario when the jobseeker want to view their job application details.
Data Return	Pop up objects containing strings
Trigger	More than one application of jobseeker exist.
Pre-conditions	Users must register as jobseeker and is logged in.
Post-conditions	Application details will pop up as a form
Basic Flow	1. The jobseeker view the list of jobs applied. Jobs will include job.title, job.Industry, job.experience, job.basic_pay, job.location, job.job_desc. 2. Jobseeker can click on the job to see more details 3. The job details will pop up as well as application status

Name	Job application
Description	This use case describes the scenario when the jobseeker want to apply for a job.
Data Return	None
Trigger	The jobseeker press apply in the job page
Pre-conditions	Jobseeker has already found his or her desired job.
Post-conditions	The job application is added to the query.
Basic Flow	1. The user found a job 2. The user click to see the job description. 3. The user click on apply button. 4. The system will add the job application.

Name	Withdraw job application
Description	This use case describes the scenario when the jobseeker want to withdraw their job application .
Data Return	None
Trigger	The jobseeker press withdraw in job application details.
Pre-conditions	One application or more have to exist.
Post-conditions	The job application is removed.
Basic Flow	1. The user click to see application details. Details will include application.status, application.date_applied, application.apply_id 2. The user click on withdraw 3. The system will remove the job application

Name	Post Jobs
Description	Employer posts a job vacancy.
Data return	None
Trigger	Employer selects the option to post a job
Pre-conditions	Employer is logged into their account
Post-conditions	Job is successfully posted on the web app
Basic Flow	1. Employer selects the option to post a job. 2. Employer fills out the job details such as job title, description, requirements, etc. (job.title, job.location, job.vac_no, job.basic_pay, job.experience, job_Industry) 3. Employer submits the job posting. 4. System validates the job posting and adds it to the database.
Alternate Flow	1. At step 4, if the required fields are not filled, the system prompts the employer to complete all required fields before submitting the job posting.

Name	Delete Jobs Post
Description	Employer deletes a job posting.
Data return	None
Trigger	Employer selects the option to manage job postings
Pre-conditions	Employer is logged into their account
Post-conditions	Job posting is removed from the system
Basic Flow	1. Employer selects the option to manage job postings. 2. Employer selects the job posting they want to delete. 3. Employer confirms the deletion. 4. System removes the job posting from the database.

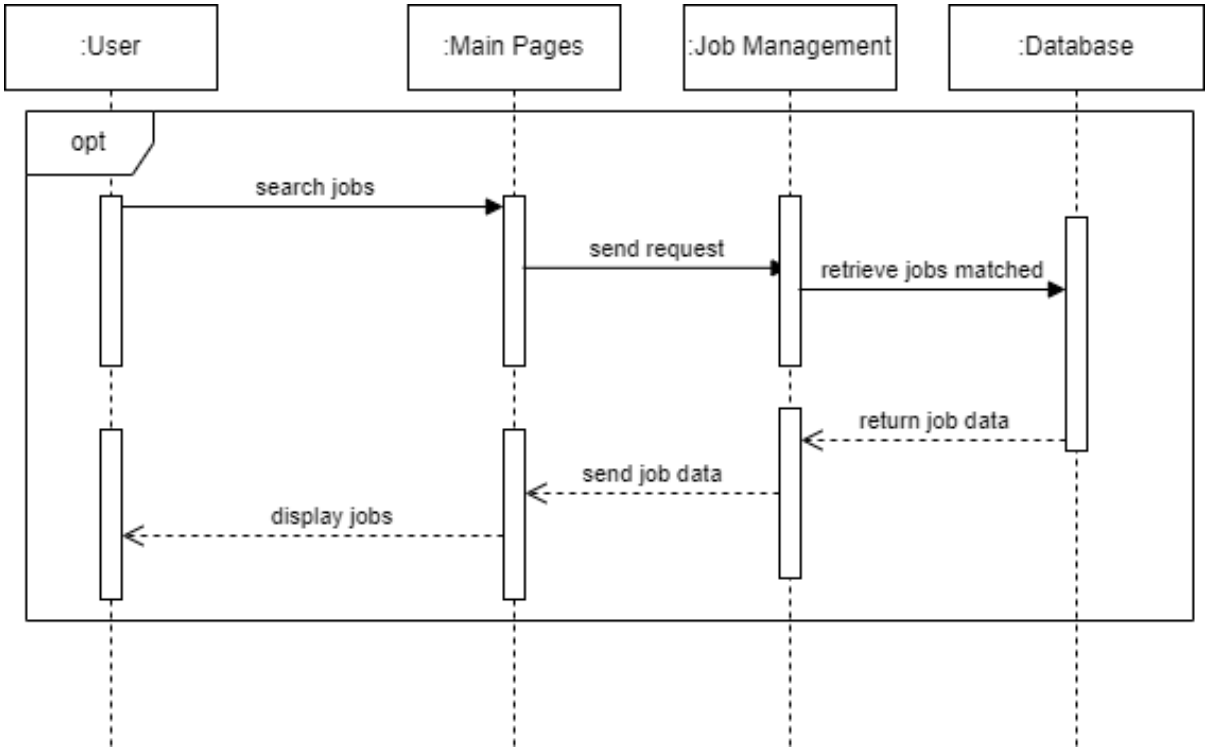
Name	View Jobs
Description	Employer views the list of jobs they have posted.
Data return	Pop up objects containing strings
Trigger	Employer navigates to the section to view posted jobs
Pre-conditions	Employer is logged into their account
Post-conditions	Employer sees the list of jobs they have posted
Basic Flow	1. Employer navigates to the section to view posted jobs. 2. System retrieves and displays the list of jobs posted by the employer. 3. Employer chooses to view a specific job. 4. System retrieves and displays the specified job details.

Name	View Applicants Info
Description	Employer views information about applicants who applied to their job postings
Data return	Pop up objects containing strings
Trigger	Employer selects a specific job posting
Pre-conditions	Employer is logged into their account
Post-conditions	Employer sees the information about applicants
Basic Flow	1. Employer selects a specific job posting. Job posting include information such as (job.title, job.location, job.vac_no, job.basic_pay, job.experience, job_Industry) 2. System retrieves and displays the list of applicants for that job posting. 3. Employer can view individual applicant details.

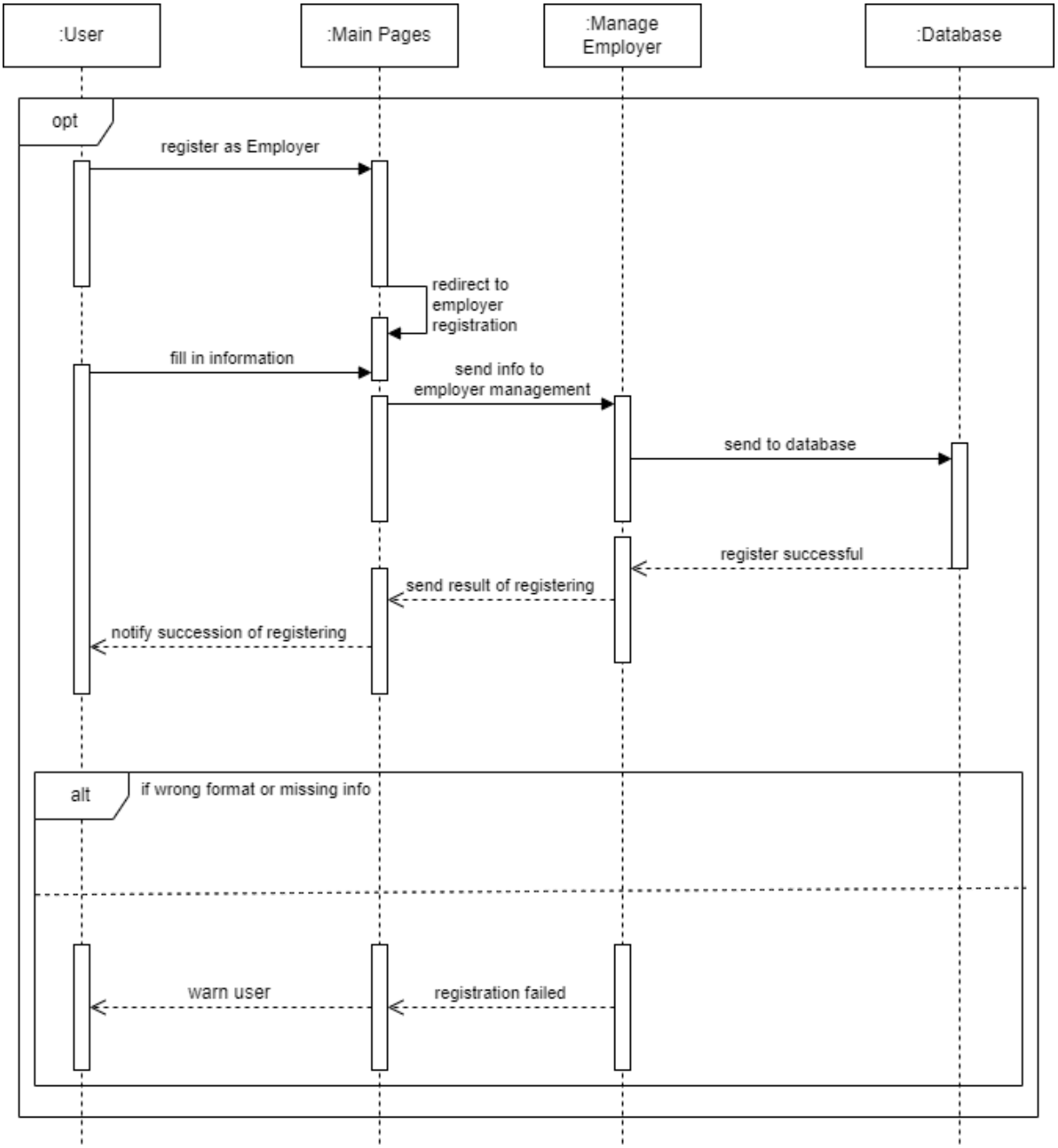
2.3 Sequence Diagram

In this section, we will present sequence diagrams respective of the aforementioned use case tables

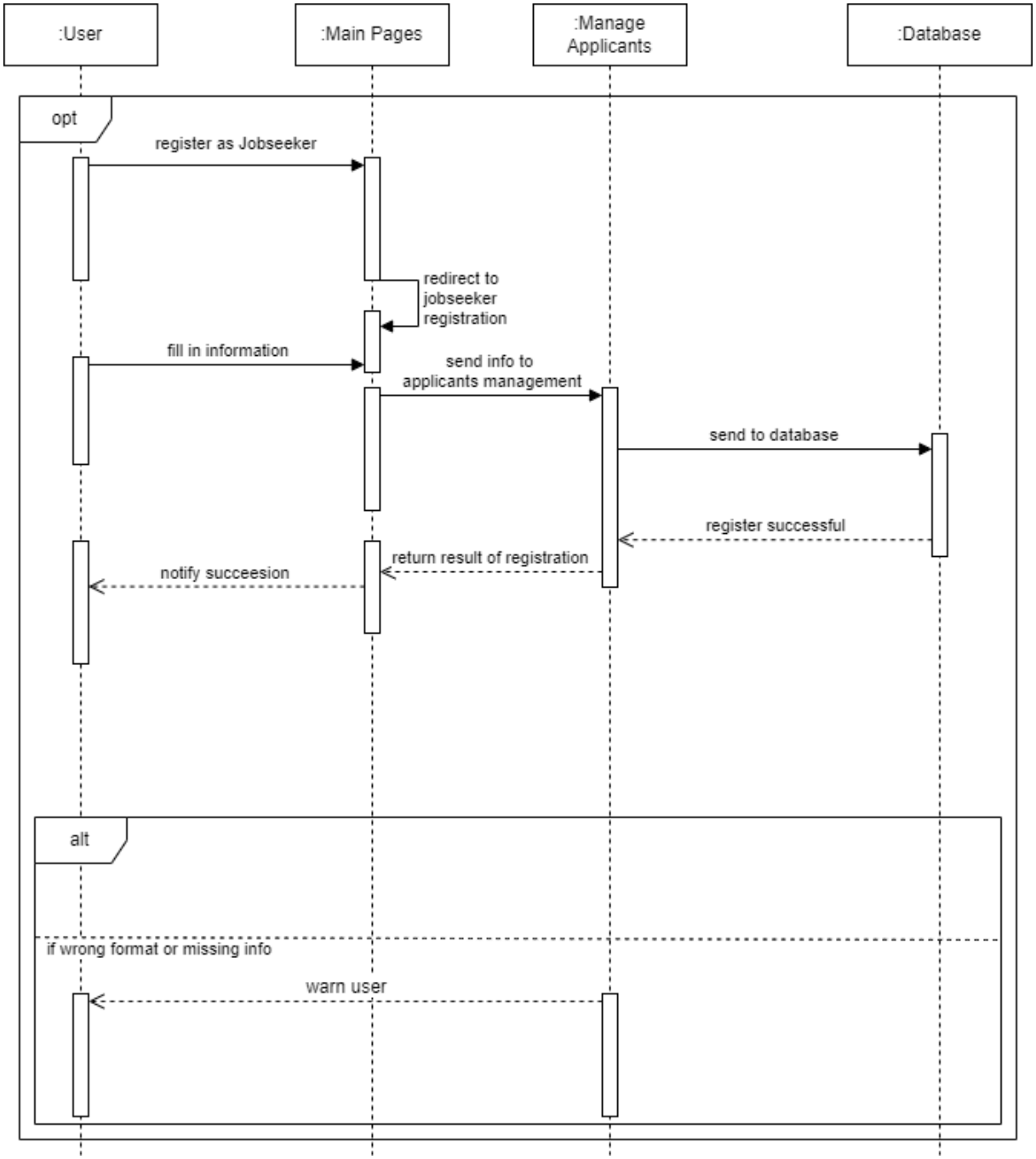
2.3.1 Search Jobs



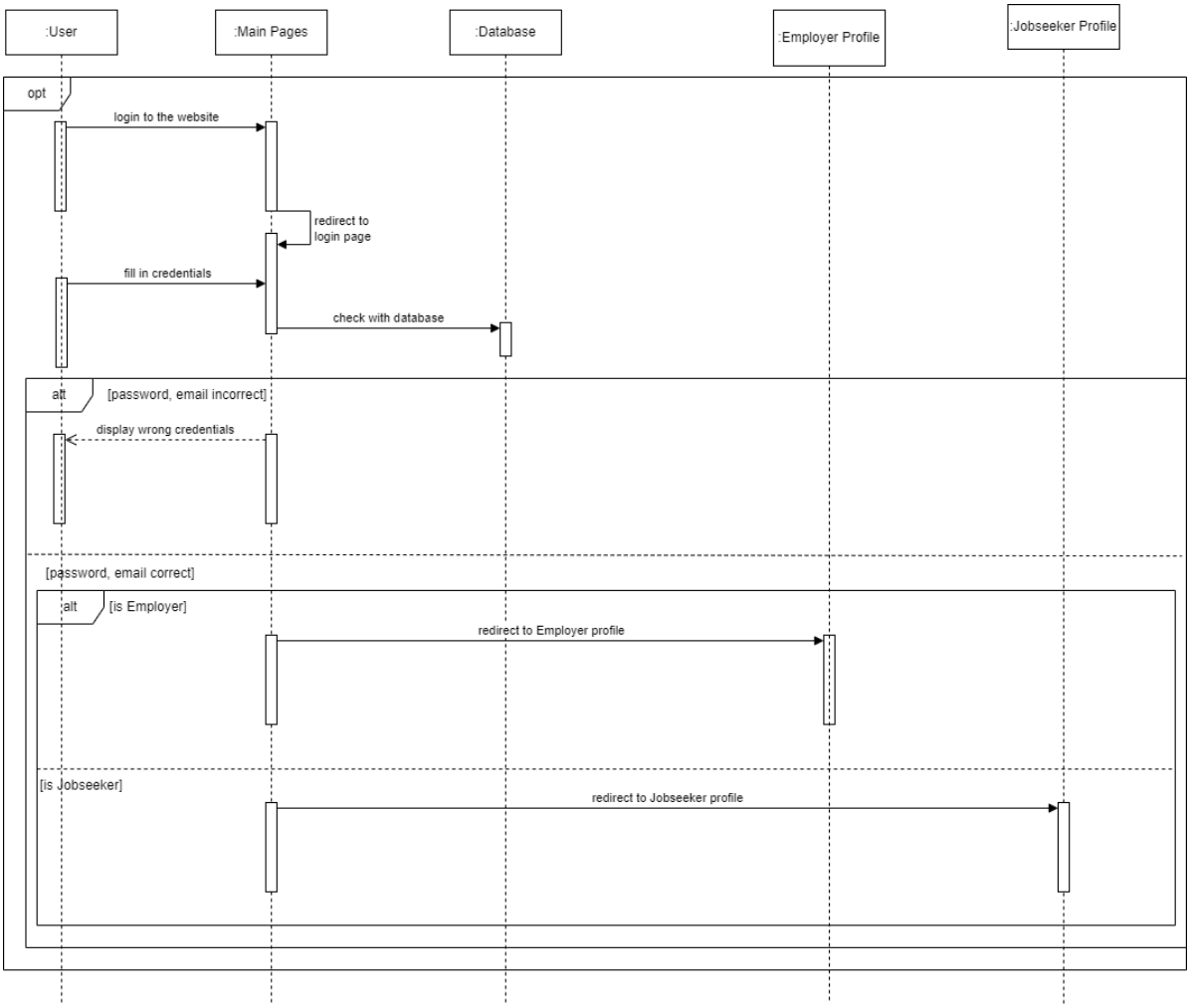
2.3.2 Register as Employer



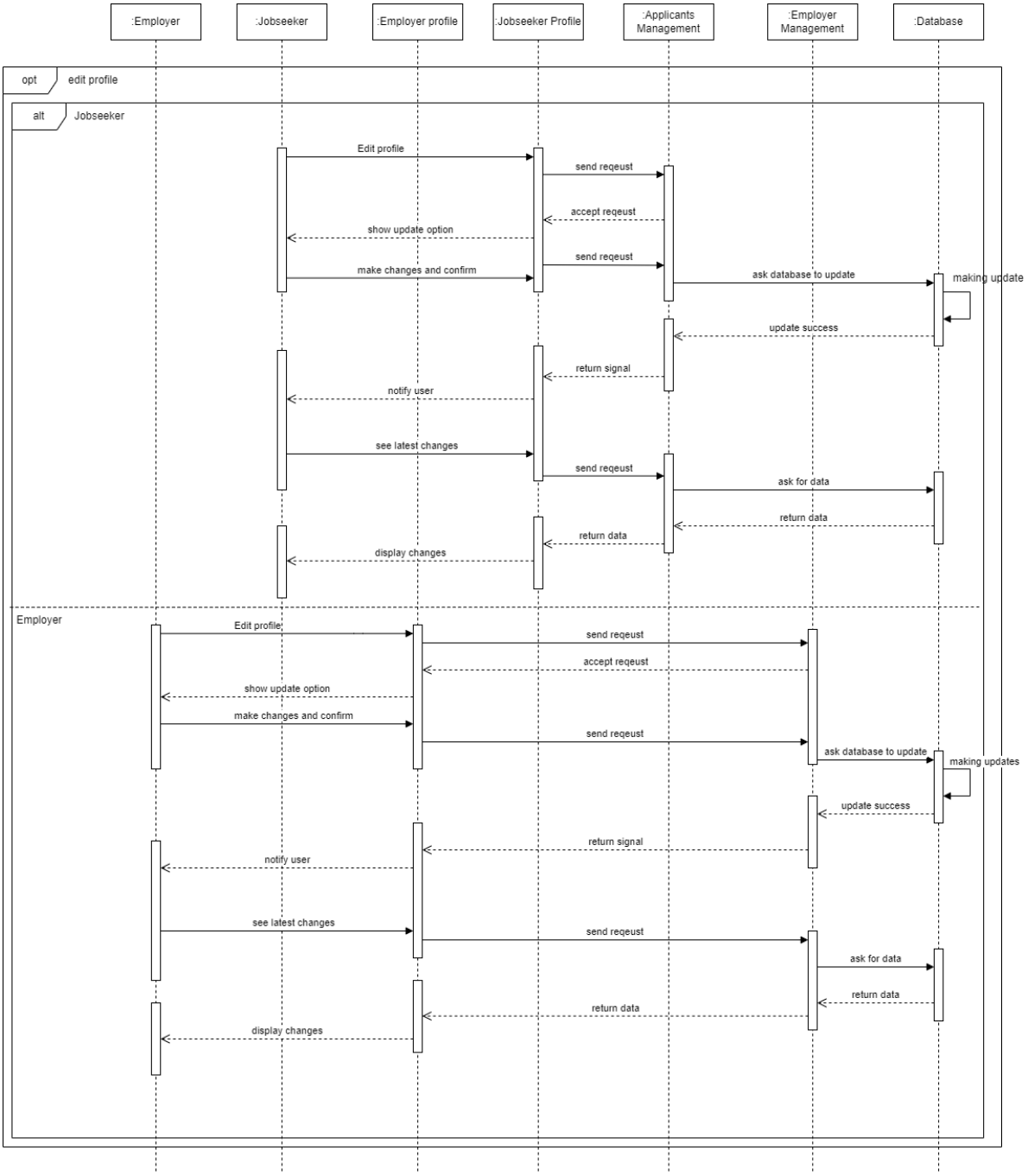
2.3.3 Register as Jobseeker



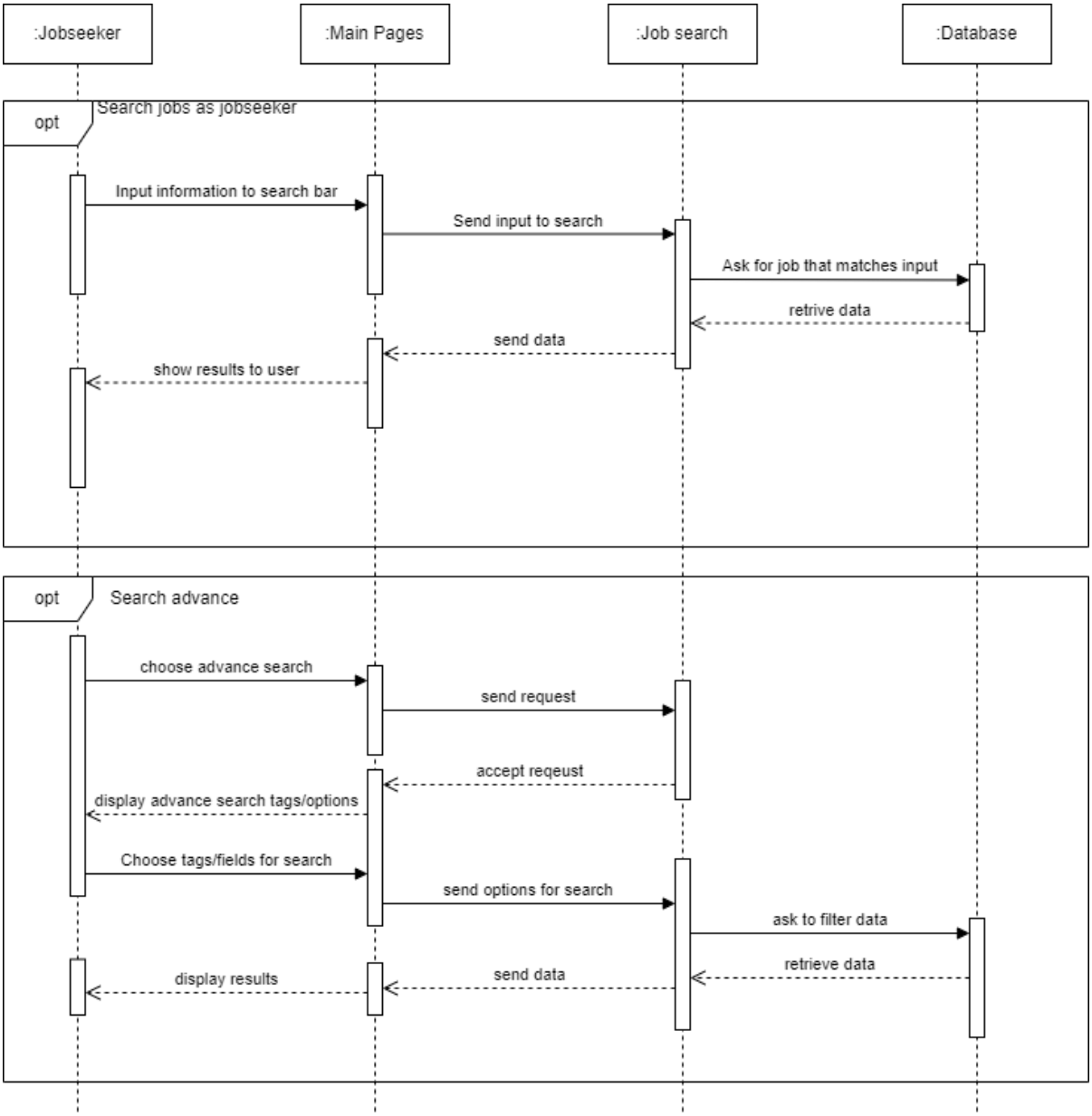
2.3.4 Login to the website



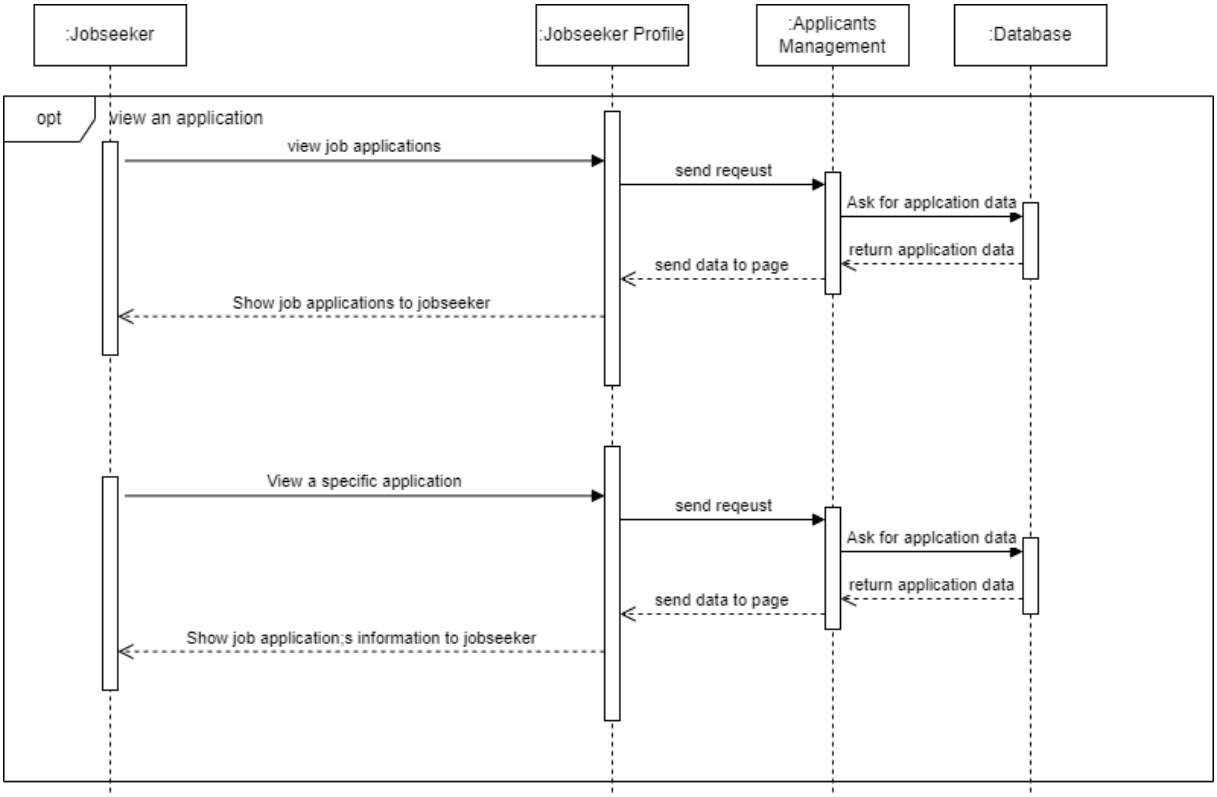
2.3.5 Edit profile



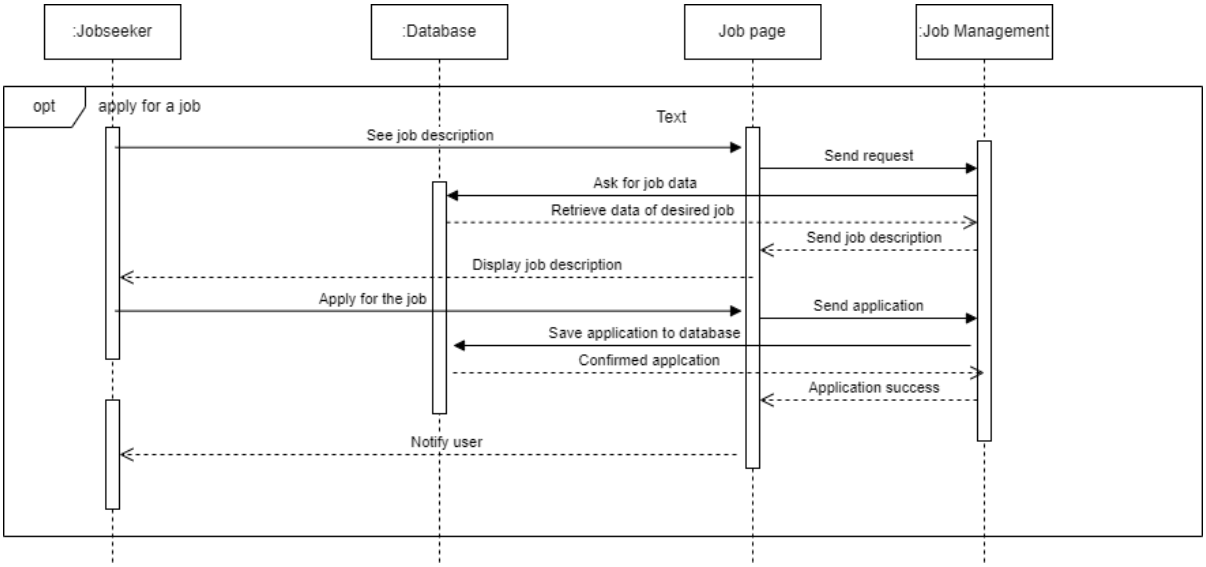
2.3.6 Search Jobs for registered user



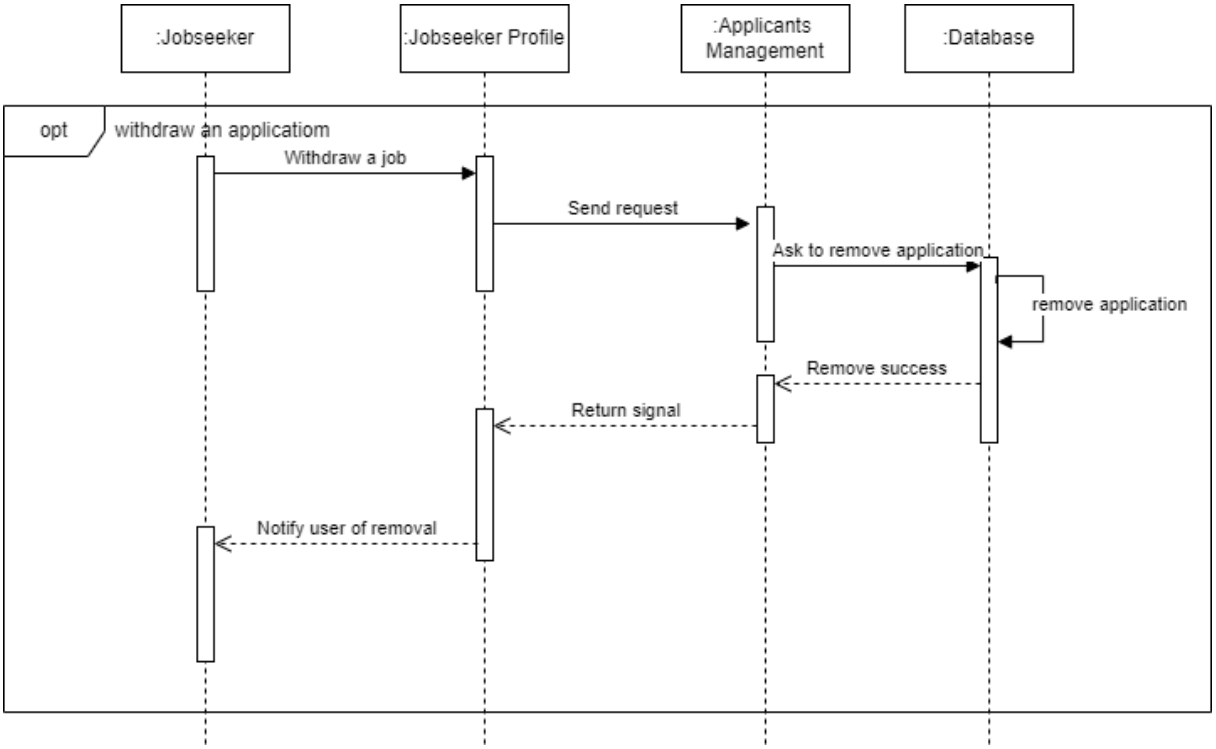
2.3.7 View jobs application details



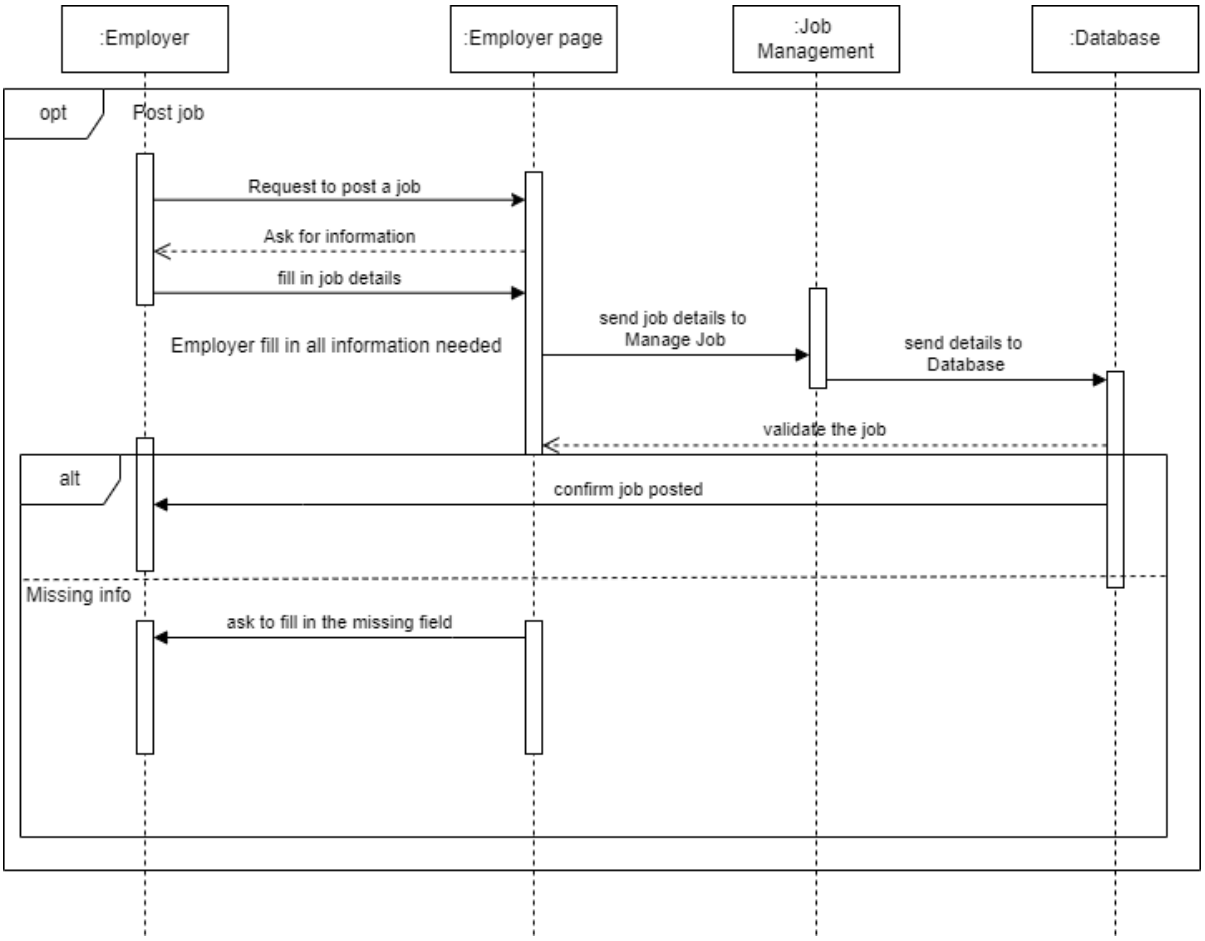
2.3.8 Job application



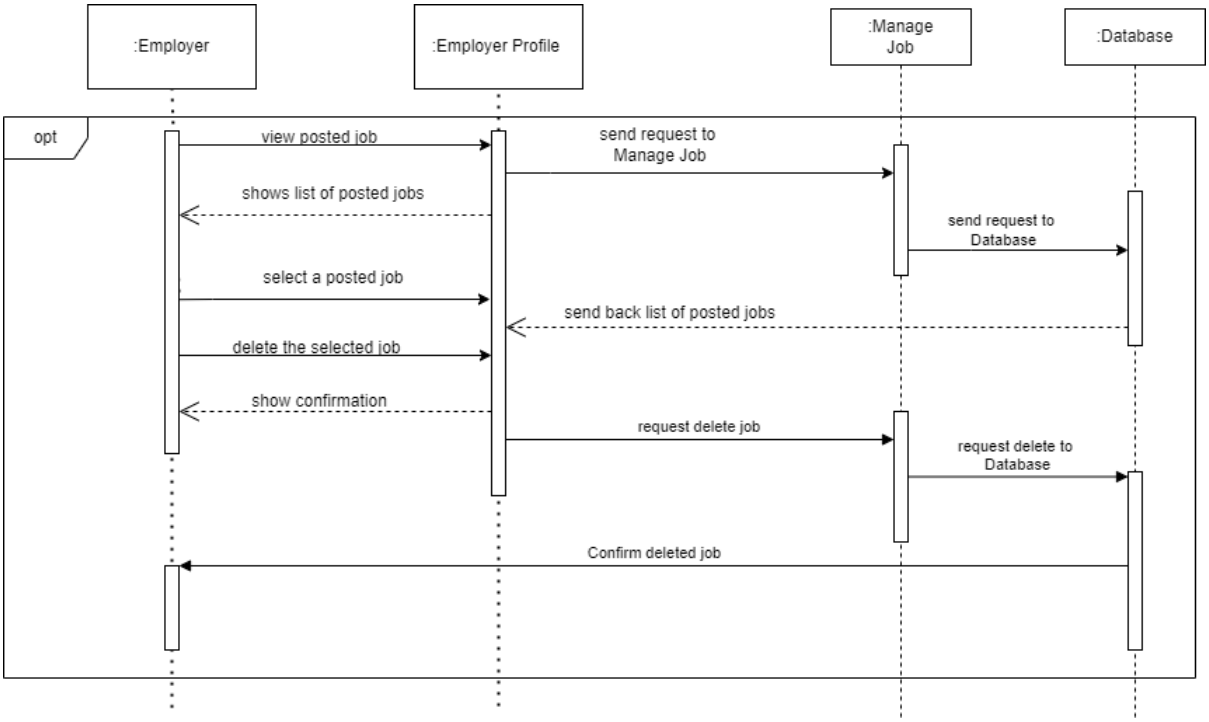
2.3.9 Withdraw job application



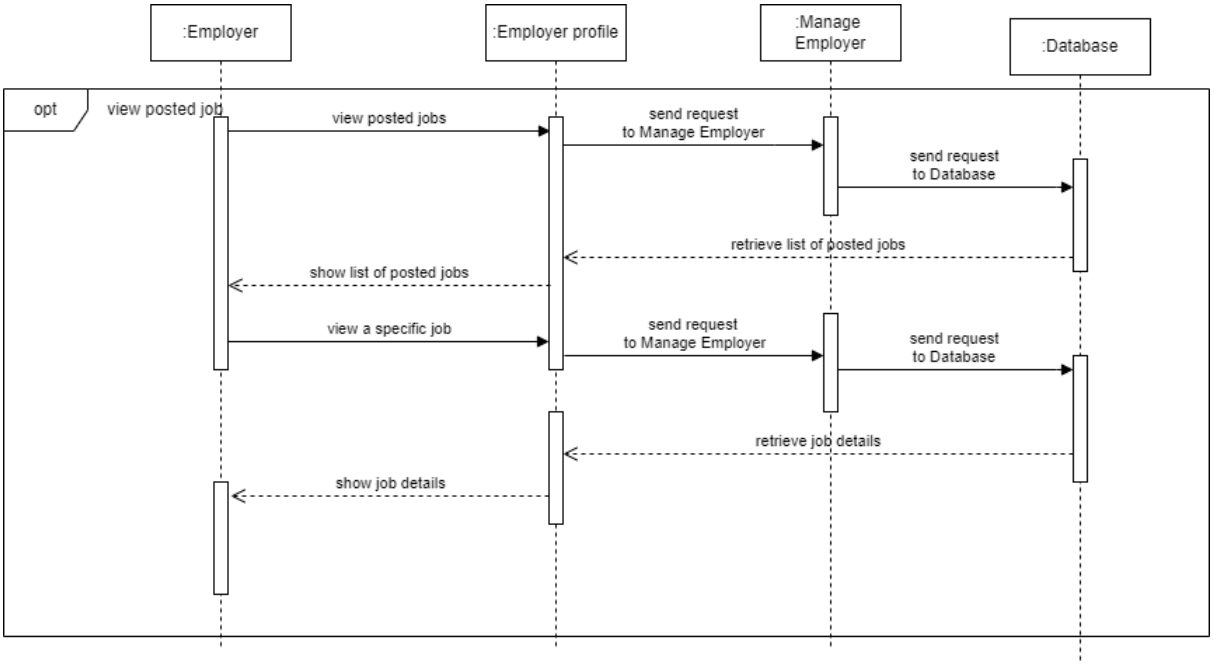
2.3.10 Post jobs



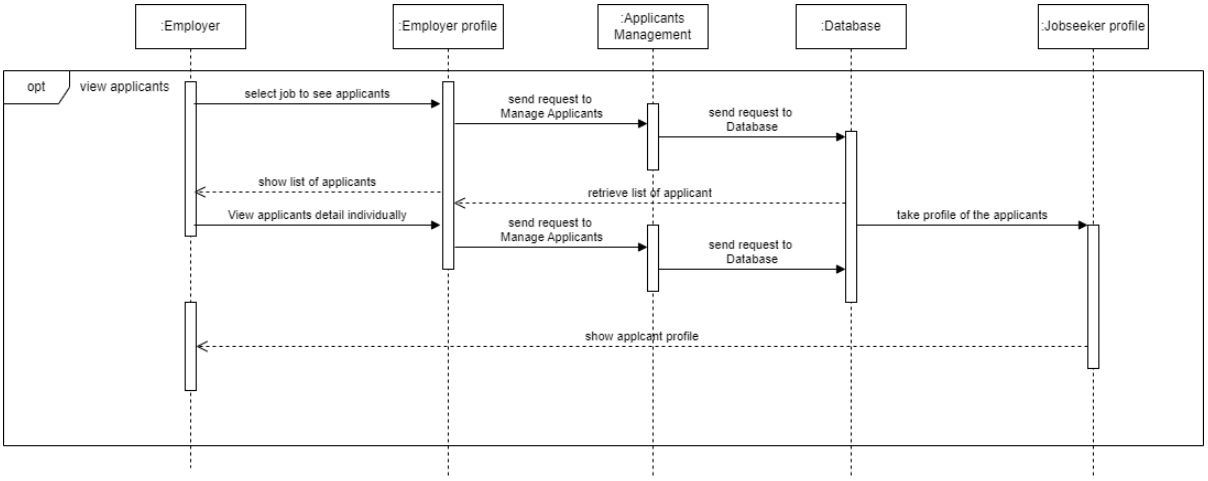
2.3.11 Delete jobs



2.3.12 View jobs

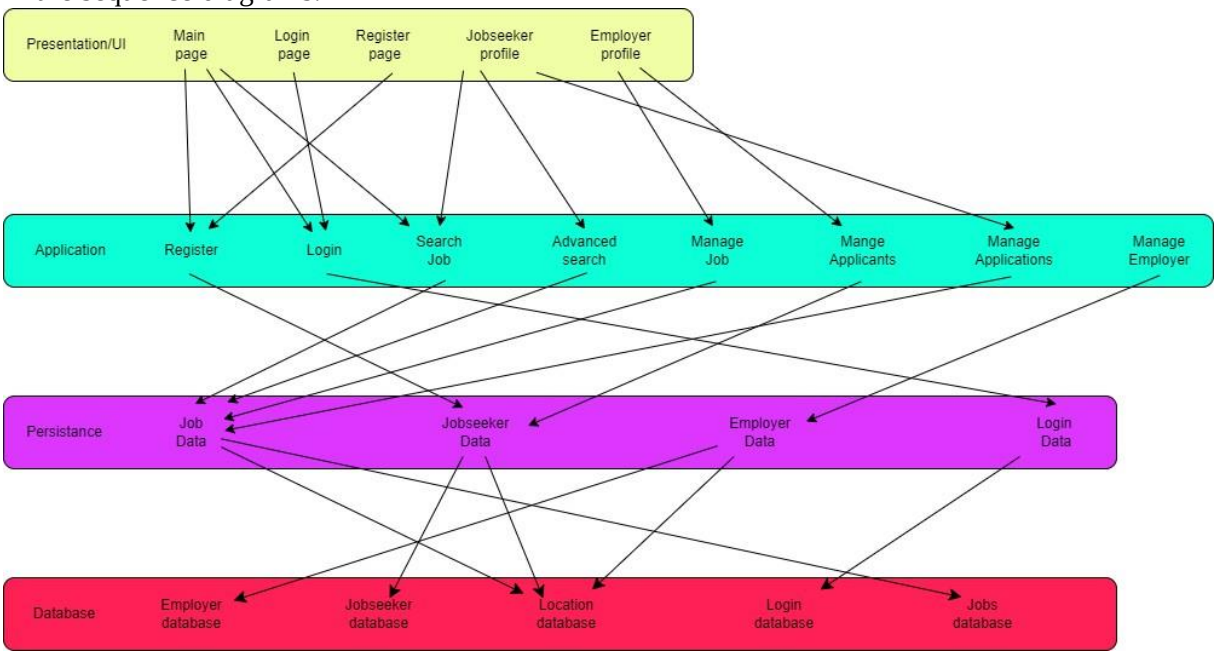


2.3.13 View applicants info



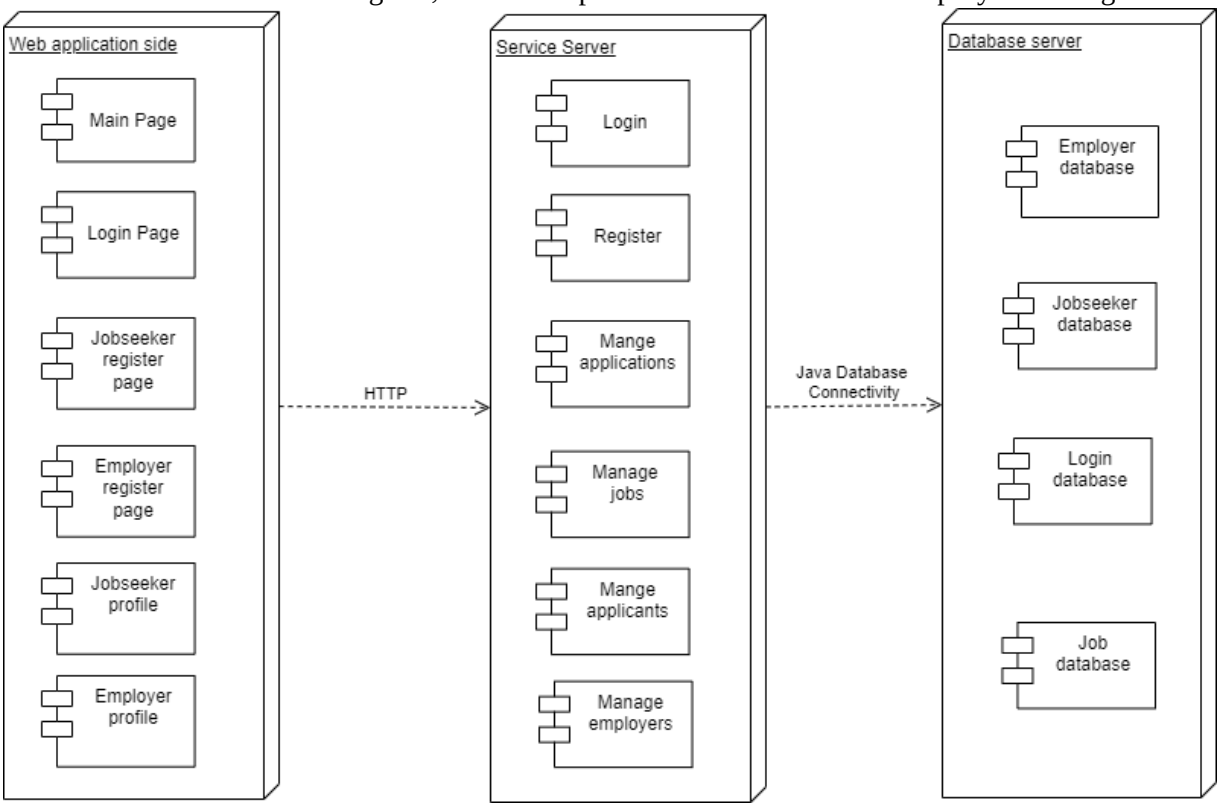
2.4 Software Architecture

Here we will present our software architecture corresponding to the components and functions in the sequence diagrams.



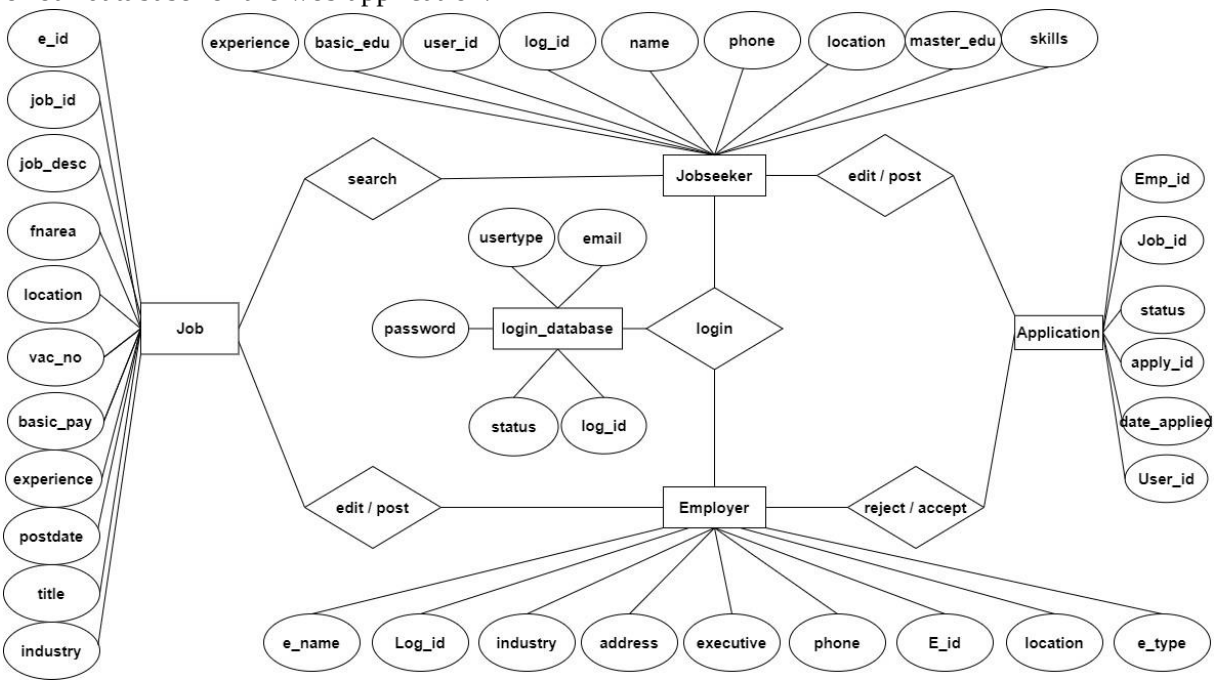
2.5 Deployment Diagram

Based on the architecture diagram, we can implement as well an UML Deployment diagram.



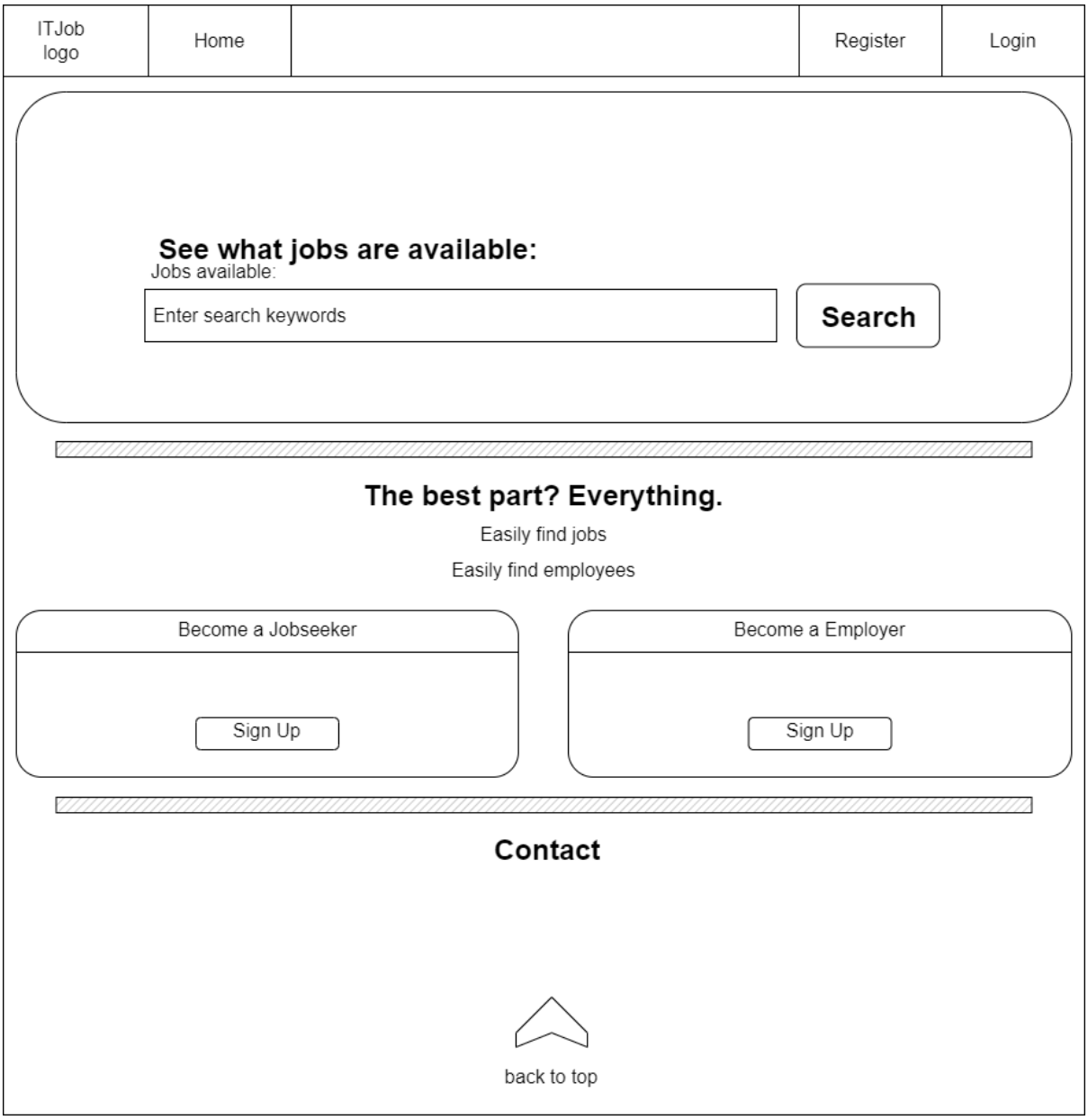
2.6 Entity-Relationship Diagram

In this section, we will attempt to brainstorm an entity relationship diagram to show the schema of our database for the web application.

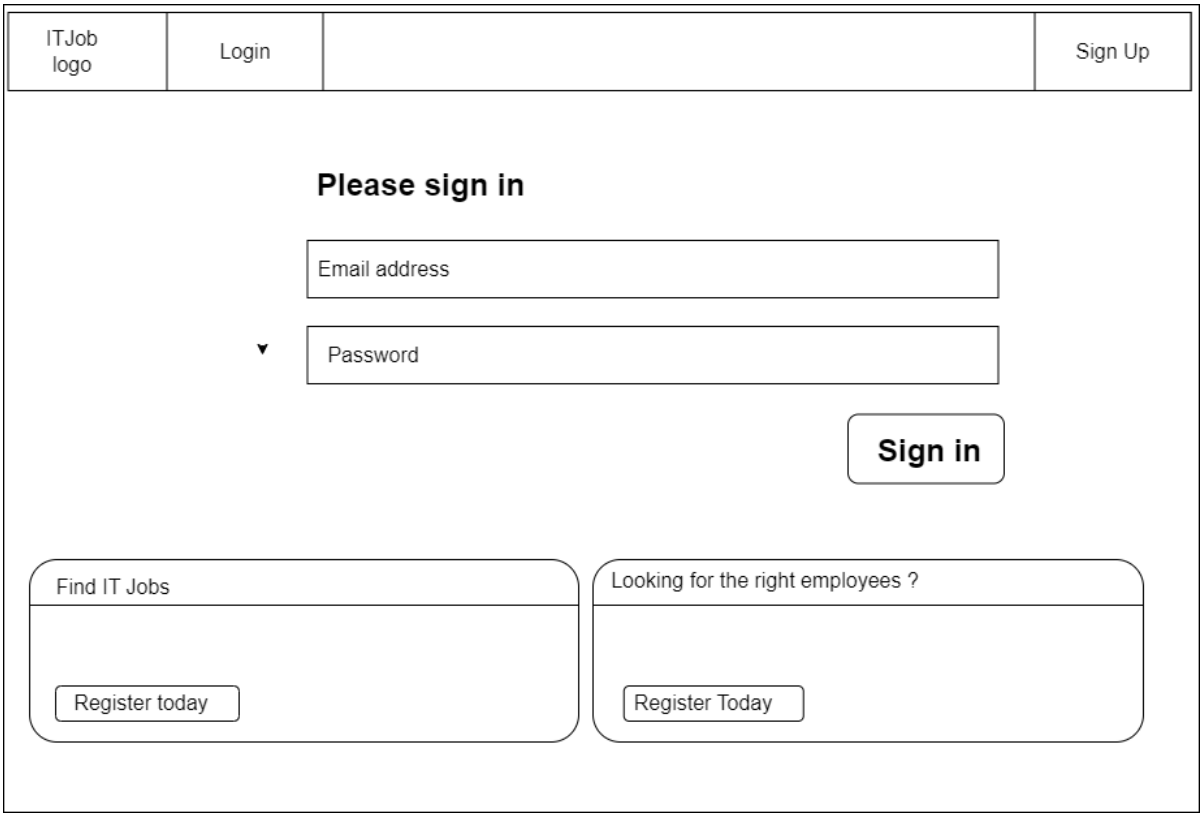


2.7 User Interface Mock-up

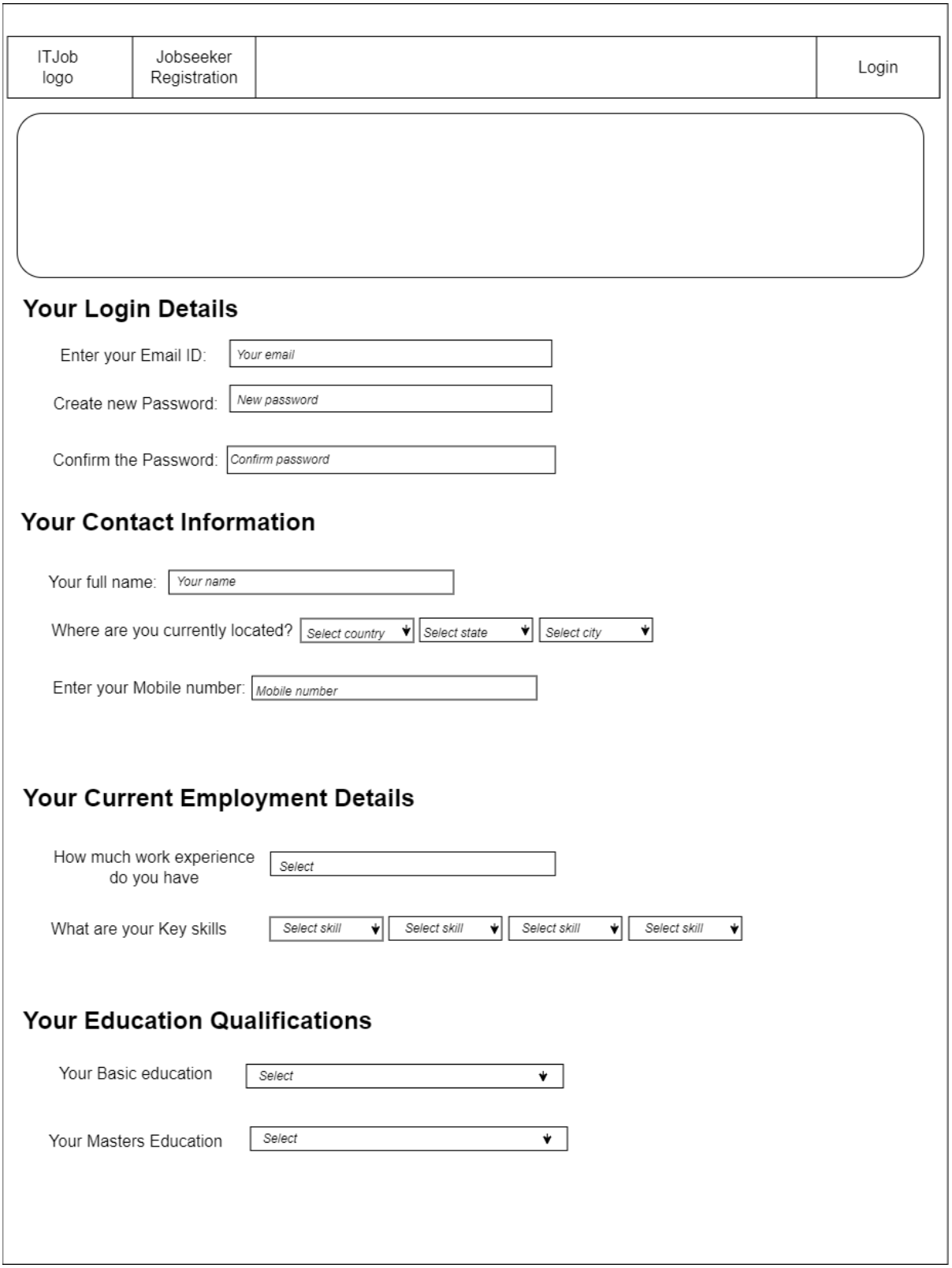
2.7.1 Main Page



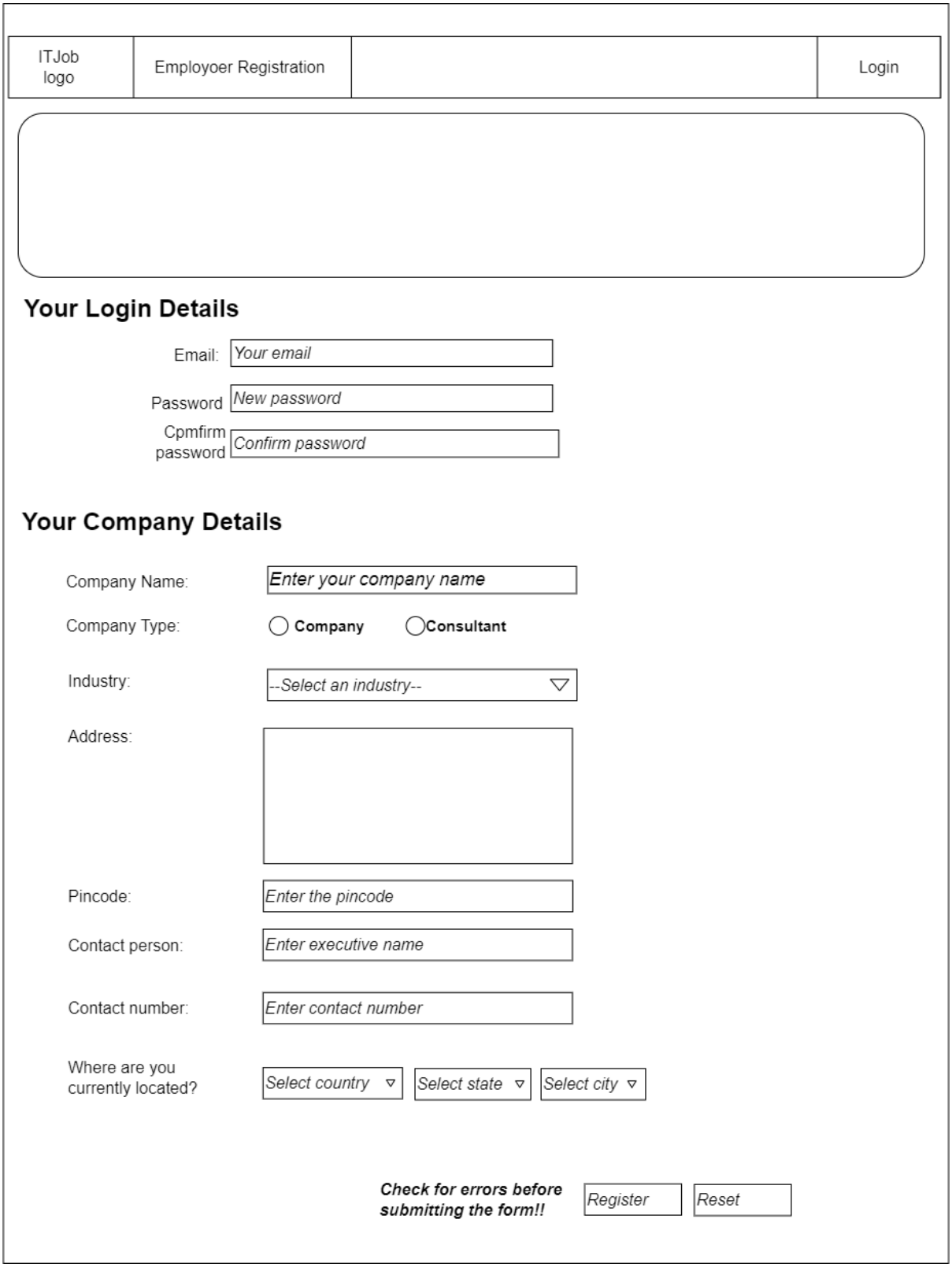
2.7.2 Login Page



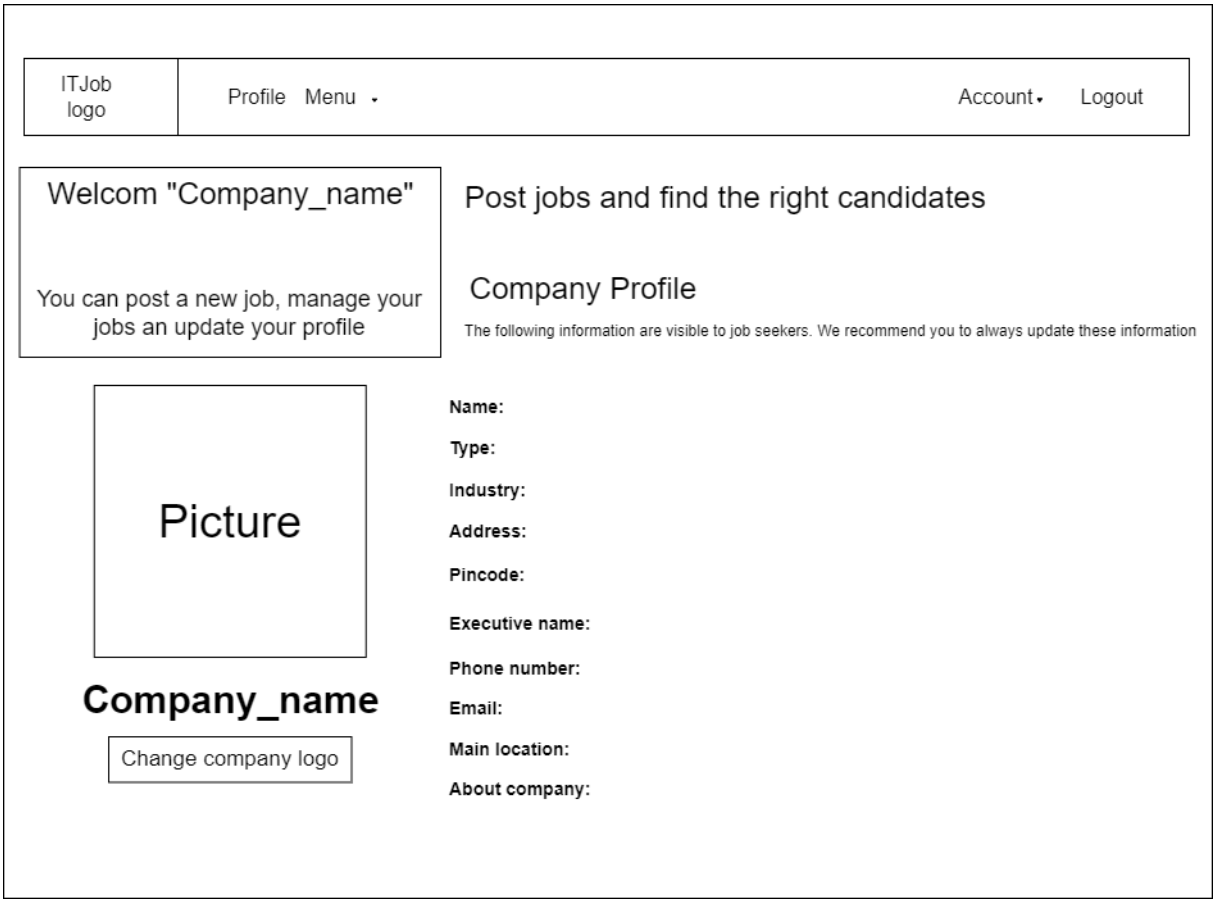
2.7.3 Jobseeker Registration Page



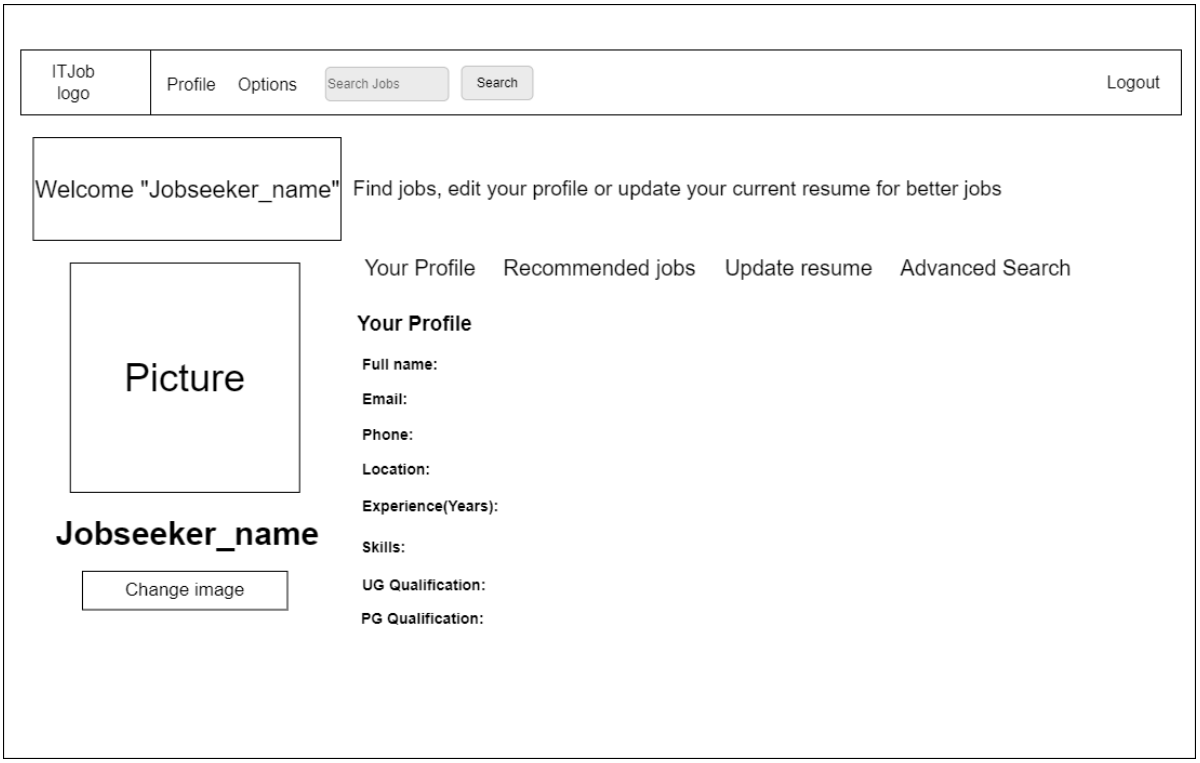
2.7.4 Employer Registration Page



2.7.5 Employer Page



2.7.6 Jobseeker Page



3 Back-end Programming

3.1 Login

```
$email= $_POST ['email'];
$passd = $_POST ['password'];
```

Retrieve User Input: The script starts by retrieving the user's email and password from the POST request. These values are stored in the '$email' and '$passd' variables, respectively.

```
$query=mysqli_query($db1,"select * from login where email='$email'");
```

Database Query: It then performs a database query to find a user in the login table with the provided email. The result of this query is stored in the $query variable.

```
$result= mysqli_fetch_array ( $query , MYSQLI_ASSOC );
```

Fetch Query Result: The script fetches the result of the query as an associative array using "mysqli_fetch_array" and stores it in the "$result" variable.

```
if(($result>0) && ( password_verify( $passd, $result['password'] ) ) ){
```

Verify User and Password: The script checks if the query returned a result and if the provided password matches the hashed password stored in the database using "password_verify".

```
if( $result['usertype']=="jobseeker")
{
    session_start ();
    $_SESSION["id"] = $result['log_id'];
    $_SESSION["type"]  =  $result['usertype'];
    header('location:jobseeker/profile.php?msg=success');
}
elseif( $result['usertype']=="employer")
{
    session_start ();
    $_SESSION["elogid"] = $result['log_id'];
    $_SESSION["type"] = $result['usertype'];
    $_SESSION ["status"]= $result['status'];
    header('location:employer/profile.php?msg=success');
}
```

User Type Handling: If the user is found and the password is correct, the script checks the user type.

If the user is a jobseeker, it starts a new session, sets session variables for the user ID and type, and redirects the user to the jobseeker profile page with a success message.

If the user is an employer, it also starts a new session, sets session variables for the user ID, type, and status, and redirects the user to the employer profile page with a success message.

```
else
{
header('location:login.php?msg=failed');
}
```

Login Failure: If the user is not found or the password does not match, the script redirects the user back to the login page with a failed message.

3.2 Register

3.2.1 Register for Employer

```
include_once('../config.php');
$email=$_POST['email'];
$password=$_POST['pass1'];
$hash = password_hash($password, PASSWORD_DEFAULT);
$name=$_POST['compname'];
$type = $_POST['comtype'];
//echo $type;
$industry=$_POST['indtype'];
//echo $industry;
$addr=$_POST['addr'];
$pin=$_POST['pin_code'];
$person=$_POST['person'];
$phone=$_POST['phone'];
$countryid=$_POST['country'];
$stateid=$_POST['state'];
$cityid=$_POST['city'];

mysqli_select_db($db2,"location");

$query1=mysqli_query($db2,"select name from countries WHERE id = '$countryid'")
    or die("Wrong Query");
$row = mysqli_fetch_assoc($query1);
$country= $row['name'];

$query2=mysqli_query($db2,"select name from states WHERE id = '$stateid'") or
    die("Wrong Query");
$row = mysqli_fetch_assoc($query2);
$state= $row['name'];
//echo $state;

$query3=mysqli_query($db2,"select name from cities WHERE id = '$cityid'") or
    die("Wrong Query");
$row = mysqli_fetch_assoc($query3);
$city= $row['name'];

$location=$country.",".$state.",".$city;

```

```
$query4 = "INSERT INTO login (email,password,usertype,status) VALUES ('$email','$hash','employer',0)";
    $result1 = mysqli_query($db1,$query4) or die("Cant Register , The user email may be already existing");
$query5 = "INSERT INTO employer (log_id,ename,phone,location,etype,address,pincode,executive,industry)
                VALUES ((SELECT log_id FROM login WHERE email='$email'),'$name','$phone','$location','$type','$addr','$pin','$person','$industry')";

 //$result2 = mysqli_query($db1,$query5);
if (!mysqli_query($db1,$query5))
{
    echo("Error description: " . mysqli_error($db1));
}
else{
    header('location:../login.php');
}
```

This PHP script, part of a registration process for employers, performs the following actions:

1. Includes the database configuration file.

2. Retrieves employer registration data (email, password, company name, company type, industry type, address, pin code, contact person, phone number, country, state, and city IDs) from a form submission via POST method.

3. Hashes the password using a default algorithm for security.

4. Selects the location database for subsequent queries.

5. Executes three queries to the location database to retrieve the names of the country, state, and city based on their IDs. Each query:

- Is executed against the db2 database connection.
- Uses the mysqli_query function to perform the SQL query.
- Utilizes or die("Wrong Query") for basic error handling, stopping script execution if a query fails.
- Fetches the result with mysqli_fetch_assoc to get the name of the location (country, state, city) from the respective tables (countries, states, cities).

6. Stores the retrieved location names in variables for further use.

3.2.2 Register as Jobseeker

```
 include_once('../config.php');
// Data retreived  begins here
$email= $_POST['useremail'];
//echo $email;
$password=$_POST['pass1'];
$hash  =  password_hash($password,  PASSWORD_DEFAULT);
```

```
//echo $password;
$name=$_POST['uname'];
 $mobile=$_POST['mobno'];
$experience=$_POST['experience'];
$skills=$_POST['skill1'].", ".$_POST['skill2'].", ".$_POST['skill3'].", ".$_POST
    ['skill4'];
$ug=$_POST['ugcourse'];
$pg=$_POST['pgcourse'];
$countryid=$_POST['country'];

 $location="";
$type="jobseeker";
// data retreived ends here

// now wants to fetch data from location db

 mysqli_select_db($db2,"location");
$query1=mysqli_query($db2,"select name from countries WHERE id = '$countryid'")
     or die("Wrong Query");
$row = mysqli_fetch_assoc($query1);
$country= $row['name'];

$query2=mysqli_query($db2,"select name from states WHERE id = '$stateid'") or
    die("Wrong Query");
$row = mysqli_fetch_assoc($query2);
$state= $row['name'];
//echo $state;

$query3=mysqli_query($db2,"select name from cities WHERE id = '$cityid'") or
    die("Wrong Query");
$row = mysqli_fetch_assoc($query3);
$city= $row['name'];
//echo $city;
 $location=$country.",".$state.",".$city;
//echo $location;
mysqli_close($db2);
 mysqli_select_db($db1,"jobportal");

$query4="INSERT INTO login (email,password,usertype,status) VALUES ('$email','
    $hash','$type',1)";
     $result1 = mysqli_query($db1,$query4) or die("Cant Register , The user email
      may be already existing");
$query5 =  "INSERT INTO jobseeker (log_id,name,phone,location,experience,skills,
    basic_edu,master_edu)
                VALUES ((SELECT log_id FROM login WHERE email='$email'),'$name
    ','$mobile','$location','$experience','$skills','$ug','$pg')";

 //$result2 = mysqli_query($db1,$query5);
if (!mysqli_query($db1,$query5))
```

```
48 {
49  echo("Error description: " . mysqli_error($db1));
50 }
51 else{
52     header('location:../login.php?msg=registered');
53 }
```

Include Configuration: The script starts by including a configuration file (config.php). This file likely contains database connection settings and other global configurations.

Retrieve Form Data: It retrieves data submitted via a POST request from a registration form. This includes the user's email, password, name, mobile number, experience, skills, under-graduate course, postgraduate course, and country ID. The password is then hashed using PHP's password_hash function for secure storage.

Concatenate Skills: The skills are received from individual fields (skill1, skill2, skill3, skill4) and concatenated into a single string, separated by commas. This suggests the form allows the user to enter up to four skills.

Set Default Values: It sets default values for location and type. The location is initially an empty string, and type is set to "jobseeker", indicating the role of the user in the system.

Database Selection for Location Data: The script selects a database named "location" using mysqli_select_db. This indicates that location-related data (countries, states) are stored in a separate database or a separate schema within the same database.

Fetch Country Name: It executes a query to fetch the name of the country based on the 'countryid' provided by the user. The country name is retrieved from the countries table. If the query fails, it terminates the script and prints "Wrong Query", which is a basic error handling mechanism.

Fetch State Name: Similar to fetching the country name, it attempts to fetch the state name using an undefined variable $stateid. This is a mistake in the script as $stateid is not defined anywhere in the provided code. This would likely result in a PHP notice or warning and could cause the query to fail or return unexpected results.

3.3 Manage applicants

```
1  function selectJs(user,job,emp) {
2              var xmlhttp;
3              if (window.XMLHttpRequest) { // for IE7+, Firefox, Chrome, Opera,
       Safari
4                  xmlhttp = new XMLHttpRequest();
5              } else { // for IE6, IE5
6                  xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
7              }
8              xmlhttp.onreadystatechange = function() {
9                  if (xmlhttp.readyState != 4 && xmlhttp.status == 200) {
10                     document.getElementById("message").innerHTML = "Processing
       Request..";
11                 } else if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
```

```
                    document.getElementById("message").innerHTML = xmlhttp.
    responseText;
                } else {
                    document.getElementById("message").innerHTML = "Error
    Occurred. <a href='manage_applicants.php'>Reload Or Try Again</a> the page."
    ;
                }
            }
            xmlhttp.open("GET", "process_select.php?user=" + user +"&job="+job +
     "&emp="+emp, true);
            xmlhttp.send();
        }

function rejectJs(user,job,emp) {
            var xmlhttp;
            if (window.XMLHttpRequest) { // for IE7+, Firefox, Chrome, Opera,
    Safari
                xmlhttp = new XMLHttpRequest();
            } else { // for IE6, IE5
                xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
            }
            xmlhttp.onreadystatechange = function() {
                if (xmlhttp.readyState != 4 && xmlhttp.status == 200) {
                    document.getElementById("message").innerHTML = "Processing
    Request..";
                } else if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
                    document.getElementById("message").innerHTML = xmlhttp.
    responseText;
                } else {
                    document.getElementById("message").innerHTML = "Error
    Occurred. <a href='manage_applicants.php'>Reload Or Try Again</a> the page."
    ;
                }
            }
            xmlhttp.open("GET", "process_reject.php?user=" + user +"&job="+job +
     "&emp="+emp, true);
            xmlhttp.send();
        }
```

Initialization of XMLHttpRequest: Both functions start by creating an XMLHttpRequest object. This object allows you to request data from a server without reloading the page. For compatibility with older versions of Internet Explorer (IE5 and IE6), an ActiveXObject is used as a fallback.

Setting up the request: The .open() method initializes a newly-created request, or re-initializes an existing one. In this case, it's configured to make a GET request to either process_select.php or process_reject.php, passing user, job, and emp as query parameters. The true parameter indicates that the request should be made asynchronously.

Handling the response: The onreadystatechange event listener is set up to handle the response

from the server. This function checks the readyState and status of the request:

- If readyState is not 4 (request finished and response is ready) and status is 200 (OK), it updates the inner HTML of an element with the ID message to "Processing Request..", indicating that the request is in progress.
- If readyState is 4 and status is 200, it updates the message element's inner HTML to display the response from the server, which could be a success message, details about the operation, or any other server-generated content.
- For any other case, it displays an error message with a link to reload the page or try again, indicating that something went wrong with the request.

Sending the request: Finally, the .send() method is called to send the request to the server.

3.4 Manage jobs

```
include_once('../config.php');
session_start();
$eid=$_SESSION['eid'];
$desig=$_POST['desig'];
$vacno = $_POST['vacno'];
$desc= $_POST['jobdesc'];
$exp = $_POST['exp'];
$money= $_POST['money'];
$salary= $_POST['pay'];
$fnarea=$_POST['fnarea'];
$countryid=$_POST['country'];
$stateid=$_POST['state'];
$cityid=$_POST['city'];
$indtype=$_POST['indtype'];
$ug=$_POST['ugcourse'];
$pg=$_POST['pgcourse'];
$profile=$_POST['profile'];
$date=date('d-m-y');
$pay=$money." ".$salary;
mysqli_select_db($db2,"location");
$query1=mysqli_query($db2,"select name from countries WHERE id = '$countryid'")
    or die("Wrong Query");
$row = mysqli_fetch_assoc($query1);
$country= $row['name'];

$query2=mysqli_query($db2,"select name from states WHERE id = '$stateid'") or
   die("Wrong Query");
$row = mysqli_fetch_assoc($query2);
$state= $row['name'];
//echo $state;

```

```
$query3=mysqli_query($db2,"select name from cities WHERE id = '$cityid'") or
    die("Wrong Query");
$row = mysqli_fetch_assoc($query3);
$city= $row['name'];

$location=$country.",".$state.",".$city;
mysqli_close ( $db2 );
mysqli_select_db($db1,"jobportal");

$query4="insert into jobs (eid,title,jobdesc,vacno,experience,basicpay,fnarea,
    location,industry,ugqual,pgqual,profile,postdate )VALUES ('$eid','$desig','
    $desc','$vacno','$exp','$pay','$fnarea','$location','$indtype','$ug','$pg','
    $profile','$date')";

if (!mysqli_query($db1,$query4))
{
 echo("Error description: " . mysqli_error($db1));
}
else{
    header('location:profile.php?msg=jobposted');
}
```

Configuration and Session Start: include_once('../config.php'); includes a configuration file that likely contains database connection settings and other global configurations. session_start(); starts a new session or resumes an existing one, allowing the use of session variables.

Retrieving Form Data: The script retrieves data submitted via a POST request from a form where an employer fills out details about a job vacancy. This includes the employer's ID ($eid from session), job designation ($desig), number of vacancies ($vacno), job description ($desc), experience required ($exp), salary details ($money and $salary), functional area ($fnarea), location details ($countryid, $stateid, $cityid), industry type ($indtype), required qualifications ($ug for undergraduate, $pg for postgraduate), job profile ($profile), and the posting date ($date).

Building the Location String: The script selects the location database to fetch the names of the country, state, and city based on their IDs. These names are concatenated to form a single string ($location) representing the job's location.

Inserting Job Details into Database: After closing the connection to the location database and selecting the jobportal database, it attempts to insert the job details into the jobs table using the mysqli_query function. The query includes all the retrieved and processed information.

Error Handling and Redirection: If the insertion fails, an error message is displayed using mysqli_error($db1). If the insertion is successful, the employer is redirected to their profile page with a message indicating that the job has been posted (header('location:profile.php msg=jobposted');).

```
include_once('../config.php');
session_start          ();
if(!isset($_SESSION['id'])){
    header('location:../login.php?msg=please_login');
```

```
}
elseif (! isset( $_GET ['jid ']){
    header('location:managejobs.php?msg=selectjob');
}
$query = "DELETE FROM jobs WHERE jobid=$_GET[jid]";
$result = mysqli_query($db1,$query);
if( $result) {
    echo "<h3 style='color: green;'> Selected Job Is Successfully Deleted</h3>";
}
else{
    echo "<h3 style='color: red;'> Failed to delete the selected job!</h3>";
}
```

Start a session: session_start(); initializes a session or resumes the current one based on a session identifier passed via a GET or POST request, or passed via a cookie.

Check if a user is logged in: The script checks if there's an id set in the session with if(!isset($_SESSION['id'])).... If not, it redirects the user to the login page with a message indicating that the user needs to log in.

Check if a job ID is provided: It then checks if a job ID (jid) is provided in the GET request with elseif(!isset($_GET['jid'])).... If not, it redirects the user to the manage jobs page with a message indicating that a job should be selected.

Delete the job: If both checks pass, it constructs a SQL query to delete the job from the jobs table where the jobid matches the provided job ID from the GET request: $query = "DELETE FROM jobs WHERE jobid=$_GET[jid]";. This line is vulnerable to SQL injection as it directly includes a GET parameter in the SQL query without sanitization or prepared statements.

Execute the query: The script then attempts to execute the query using mysqli_query($db1, $query);. It checks if the query was successful.

Feedback to the user: If the query was successful, it displays a message indicating that the job was successfully deleted. If not, it displays a message indicating failure to delete the job.

3.5 Manage applications

```
function rejectjob(jobid) {
            // alert(keyword);
            var xmlhttp;
            if (window.XMLHttpRequest) { // for IE7+, Firefox, Chrome, Opera,
    Safari
                xmlhttp = new XMLHttpRequest();
            } else { // for IE6, IE5
                xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
            }
            xmlhttp.onreadystatechange = function() {
                if (xmlhttp.readyState != 4 && xmlhttp.status == 200) {
                    document.getElementById("message").innerHTML = "Processing..
    ";
                } else if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
```

```
                    document.getElementById("message").innerHTML = xmlhttp.
    responseText;
                } else {
                    document.getElementById("message").innerHTML = "Error
    Occurred. <a href='profile.php'>Reload Or Try Again</a> the page.";
                }
            }
            xmlhttp.open("GET", "reject.php?jid=" + jobid , true);
            xmlhttp.send();
        }
```

The rejectjob function in the provided PHP file sends an asynchronous GET request to reject.php with a job ID as a query parameter. It updates the content of an element with the ID message on the webpage based on the response from the server.

If the request is in process and successful, it displays "Processing..". Once completed successfully, it shows the server's response.

If there's an error, it displays an error message with a link to reload or try again.

```
function apply(jobid) {
    // alert(keyword);
    var xmlhttp;
    if (window.XMLHttpRequest) { // for IE7+, Firefox, Chrome, Opera, Safari
    xmlhttp = new XMLHttpRequest();
    } else { // for IE6, IE5
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    }
    xmlhttp.onreadystatechange = function() {
    if (xmlhttp.readyState != 4 && xmlhttp.status == 200) {
    document.getElementById("applydiv").innerHTML = "Processing..";
    } else if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
    document.getElementById("applydiv").innerHTML = xmlhttp.responseText;
    } else {
    document.getElementById("applydiv").innerHTML = "Error Occurred. <a href='
    profile.php'>Reload Or Try Again</a> the page.";
    }
    }
    xmlhttp.open("GET", "apply_job.php?jid=" + jobid , true);
    xmlhttp.send();
    }
```

The apply function sends an asynchronous GET request to apply_job.php with a job ID (jobid) as a query parameter. It updates the content of an element with the ID applydiv on the webpage based on the response from the server. If the request is in process, it displays "Processing..". Once completed successfully, it shows the server's response. If there's an error, it displays an error message with a link to reload or try again.

3.6 Manage employers

```
// Start the session
```

```
session_start();

// Include your database connection file here
include('../config.php');


// Check if form is submitted
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $name=$_POST['ename'];
    $type=$_POST['etype'];

    //echo $type;
    $industry=$_POST['industry'];

    //echo $industry;
    $addr=$_POST['address'];
    $pin=$_POST['pincode'];
    $person=$_POST['executive'];
    $phone=$_POST['phone'];
    $countryid=$_POST['country'];
    $stateid=$_POST['state'];
    $cityid=$_POST['city'];
    $location="";

    // now wants to fetch data from location db
    mysqli_select_db($db2,"location");
    $query1=mysqli_query($db2,"select name from countries WHERE id = '$countryid
    '")   or die("Wrong Query");
    $row = mysqli_fetch_assoc($query1);
    $country= $row['name'];

    $query2=mysqli_query($db2,"select name from states WHERE id = '$stateid'")
    or die("Wrong Query");
    $row = mysqli_fetch_assoc($query2);
    $state= $row['name'];
    //echo $state;

    $query3=mysqli_query($db2,"select name from cities WHERE id = '$cityid'")
    or die("Wrong Query");
    $row = mysqli_fetch_assoc($query3);
    $city= $row['name'];
    //echo $city;
    $location=$country.",".$state.",".$city;
    //echo $location;
    mysqli_close($db2);
    mysqli_select_db($db1,"jobportal");

    // Prepare an UPDATE statement
    $sql = "UPDATE employer SET ename=?, etype=?, industry=?, address=?, pincode
```

```
    =?, executive=?, phone=?, location=? WHERE eid=?";
48
49
50      if ($stmt = mysqli_prepare($db1, $sql)) {
51          // Bind variables to the prepared statement as parameters
52          mysqli_stmt_bind_param($stmt, "ssssisssi", $name, $type, $industry,
    $addr, $pin, $person, $phone, $location, $_SESSION['eid']);
53
54          // Attempt to execute the prepared statement
55          if (mysqli_stmt_execute($stmt)) {
56              // Redirect to profile page
57              header("location: profile.php");
58              exit();
59          } else {
60              echo "Something went wrong. Please try again later.";
61          }
62      }
63
64      // Close statement
65      mysqli_stmt_close($stmt);
66  }
67
68  // Close connection
69  mysqli_close($db1);
```

Session Start: The script begins by starting a new session or resuming the current one using session_start(). This is necessary for accessing session variables, which are used for maintaining state across different pages.

Database Connection: It includes the database configuration file ../config.php to establish a connection to the database. This file likely contains database connection details and is placed one directory up from the current script.

Form Submission Check: The script checks if the form has been submitted using the POST method. This is done by checking the $_SERVER["REQUEST_METHOD"] variable.

Retrieving Form Data: Upon form submission, it retrieves the employer's profile information from the POST request. This includes the employer's name, type, industry, address, pin code, executive name, phone number, and location details (country, state, city IDs).

Fetching Location Data: The script then selects the location database using mysqli_select_db($db2,"location") and fetches the names of the country, state, and city based on the IDs provided in the form. These names are concatenated to form a complete location string.

Updating the Employer's Profile: After fetching the location details, the script switches the database context to jobportal using mysqli_select_db($db1,"jobportal"). It prepares an SQL UPDATE statement to update the employer's profile with the new data. The mysqli_prepare() function is used to prepare the SQL statement, and mysqli_stmt_bind_param() is used to bind the form data to the prepared statement's parameters.

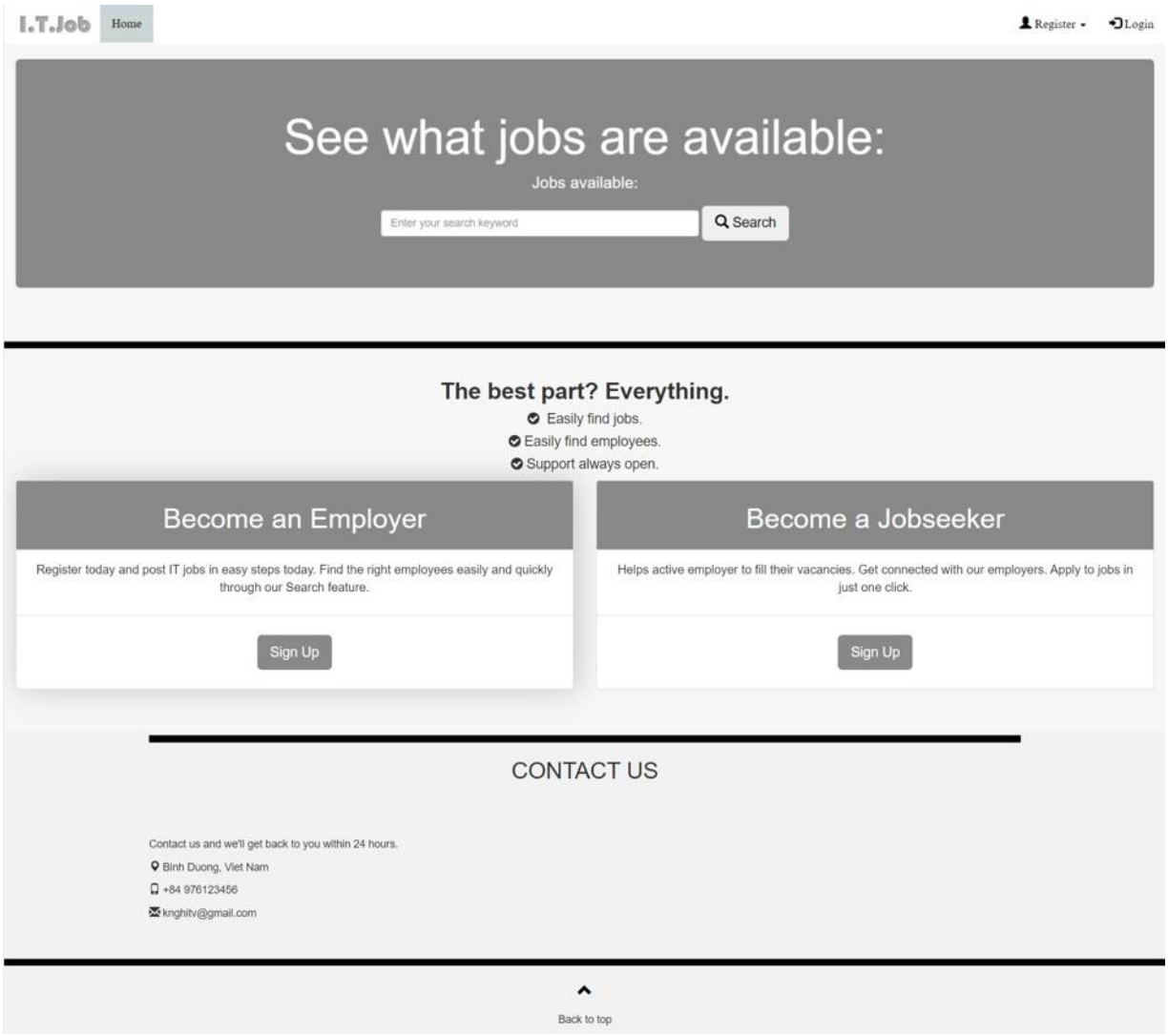
Executing the Update Statement: The script attempts to execute the prepared statement

using mysqli_stmt_execute(). If the execution is successful, it redirects the user to the profile page using the header("location: profile.php") function. If the execution fails, it displays an error message.

Closing Resources: Finally, the script closes the prepared statement and the database connection using mysqli_stmt_close($stmt) and mysqli_close($db1), respectively, to free up resources.

4 Front-end Programming

4.1 Main Page

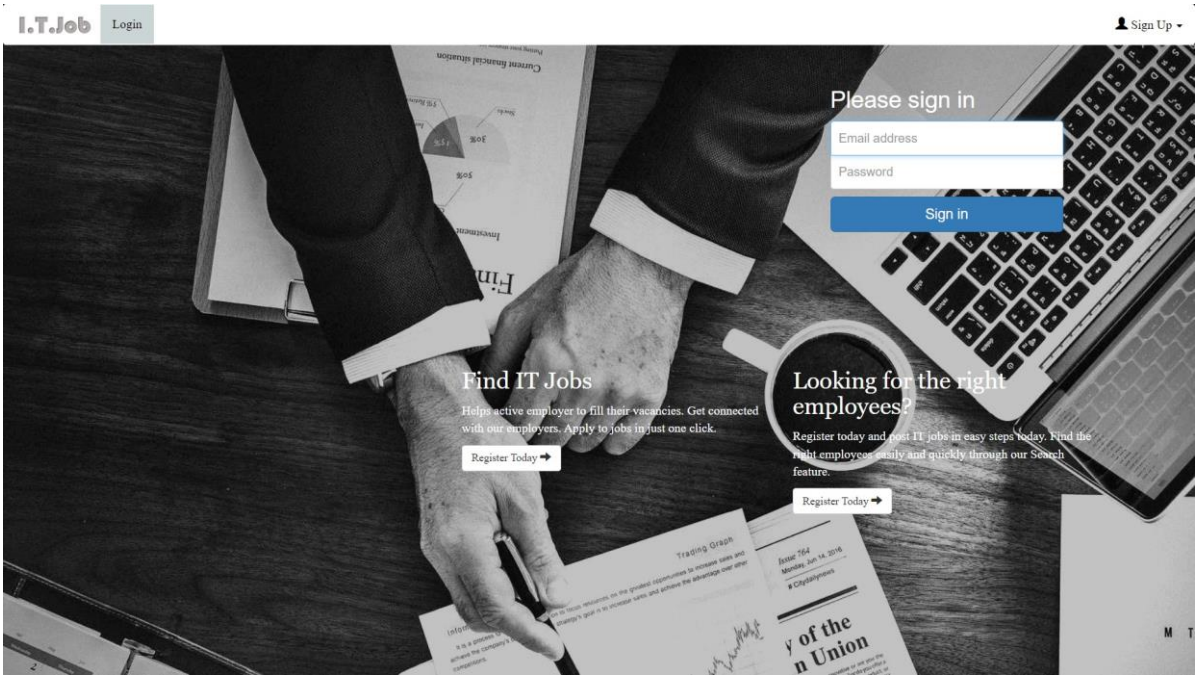


The main page has the fewest functions of all pages. Its only significance function is the search function. This function will take input from user as keyword to search for the name of the job and when user hit the search button, it will make a GET HTTP(AJAX) request to a server-side script (home_search.php) with the keyword as the parameter. The function will display "Searching.." as it executes the request in various states. Upon completion, the results or an error message will pop up Employment of the search function:

```
1 function search() {
```

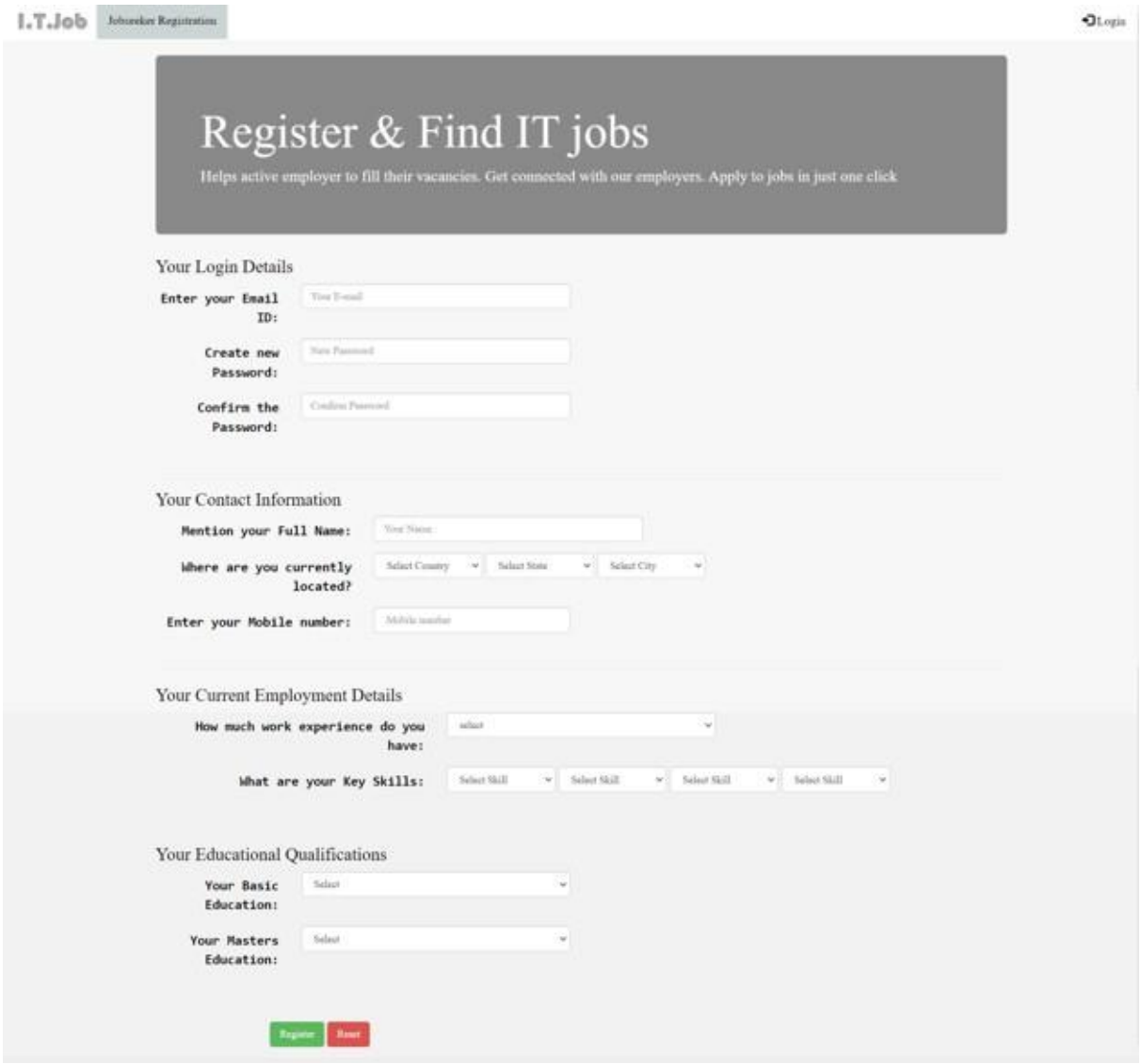
```
        var   keyword = document. getElementById (" keyword "). value ;
        // alert( keyword );
        var xmlhttp ;
        if (window .XMLHttp Request) { // for IE7+, Firefox , Chrome , Opera , Safari
            xmlhttp = new XMLHttp Request ();
        } else { // for IE6, IE5
        ...
}
```

4.2 Login Page



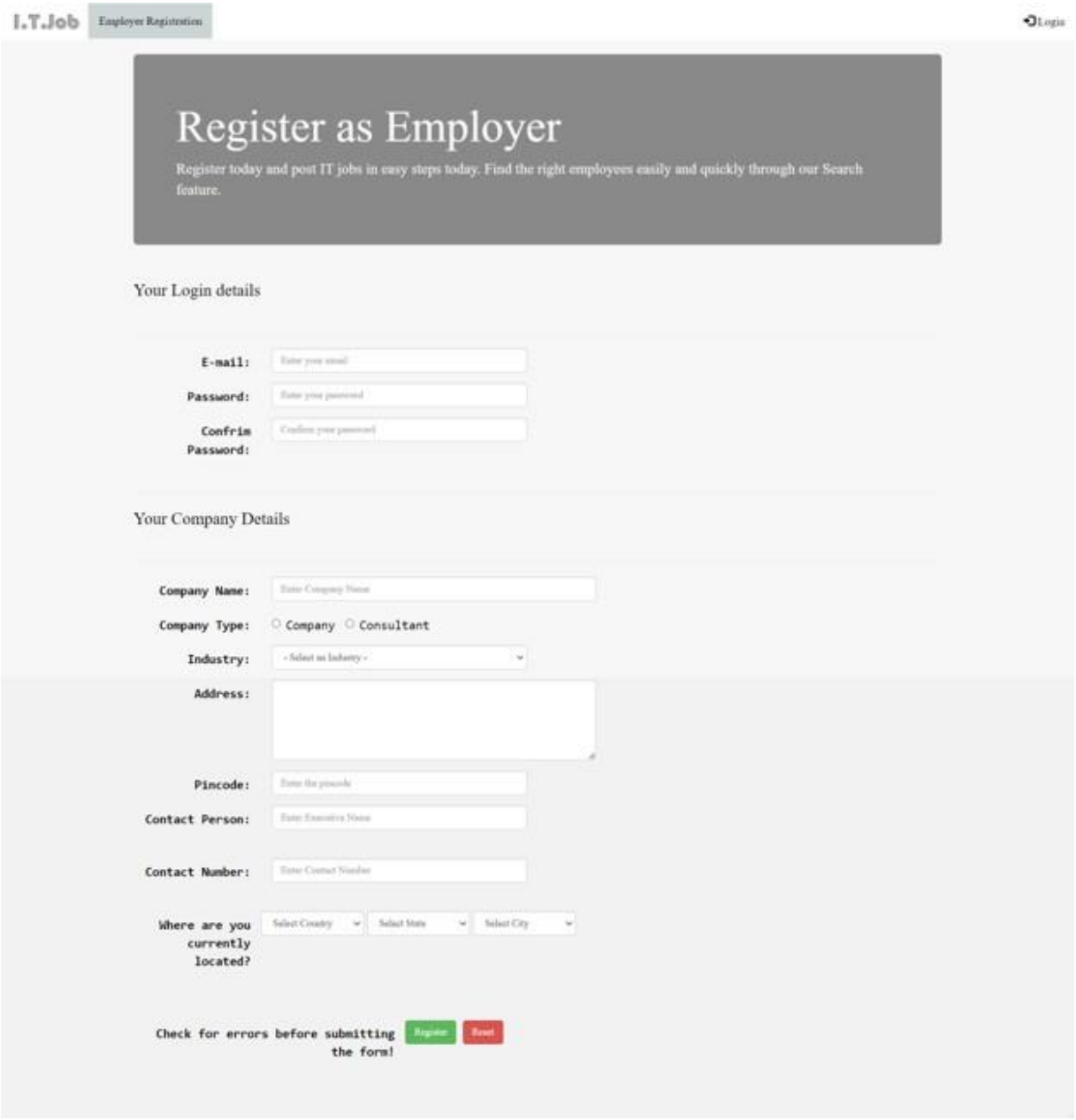
The login page, as its name states, it has the function for users to enter their credentials to access the page. Also, there are hyperlinks to go for registration. These function will be expressed later as it is related to the back-end rather than front-end

4.3 Jobseeker Registration Page



The front-end allows user to input their information such as names, phone number, level of education, etc., in order to complete their registration as well as sett up their profile. Also, there are two buttons for submitting an resetting the form. However, apart from inputting, other functions that are relevant to registration is more on the side of back-end.

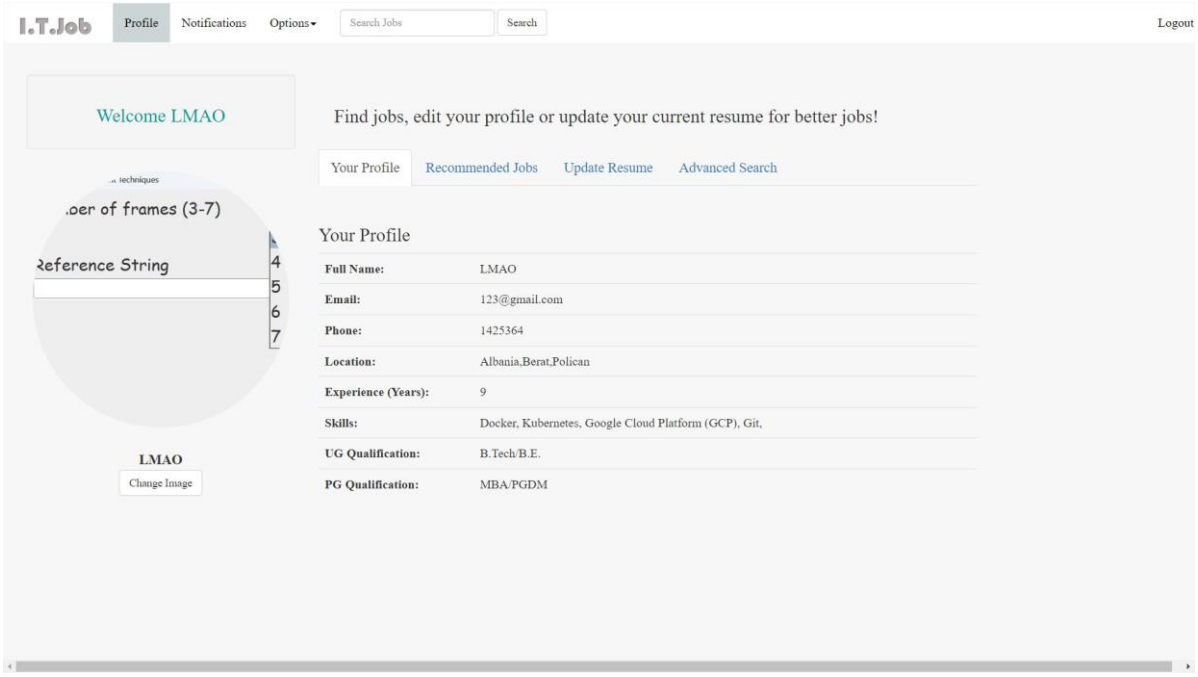
4.4 Employer Registration Page



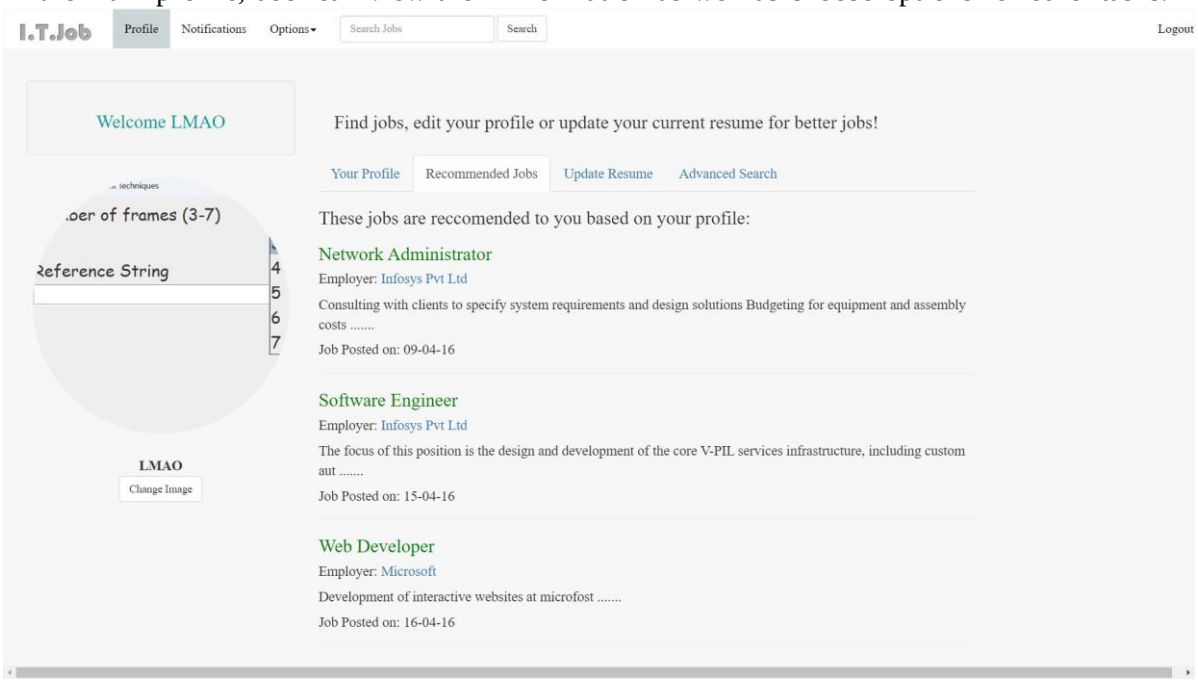
Similar to the Jobseekers' registration, the one of employers also include a form to fill in, buttons to submit and reset while the other functions lean to back-end.

4.5 Jobseeker Profile

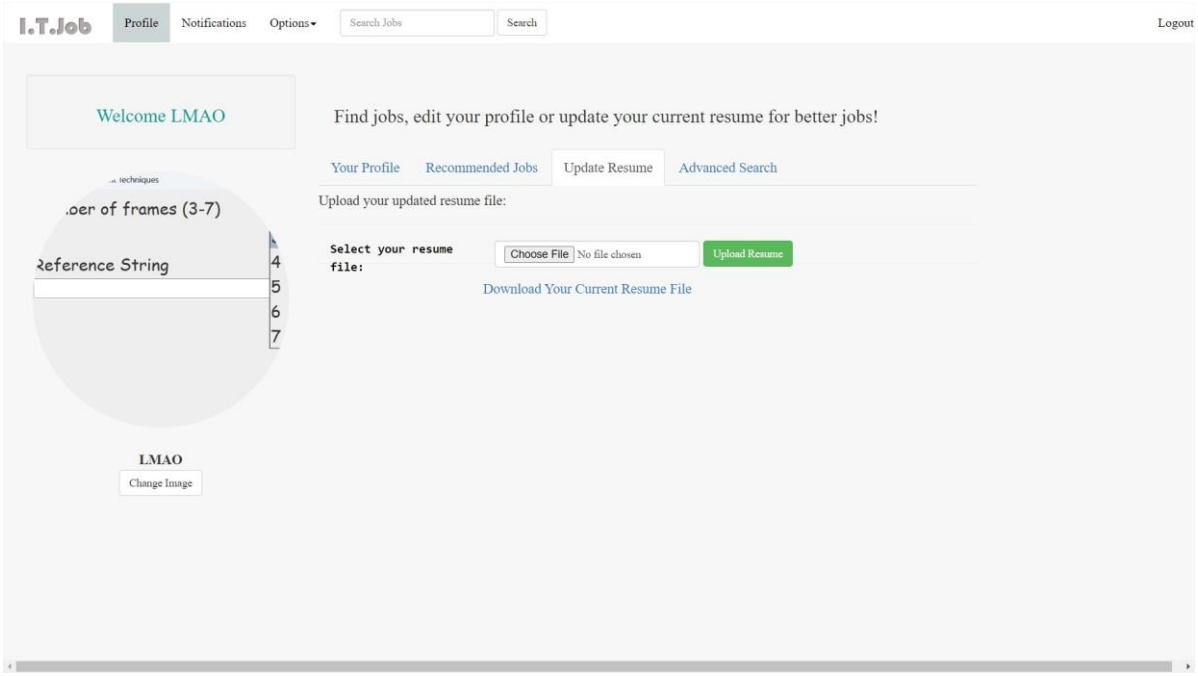
Profile of jobseekers has many function in front-end.



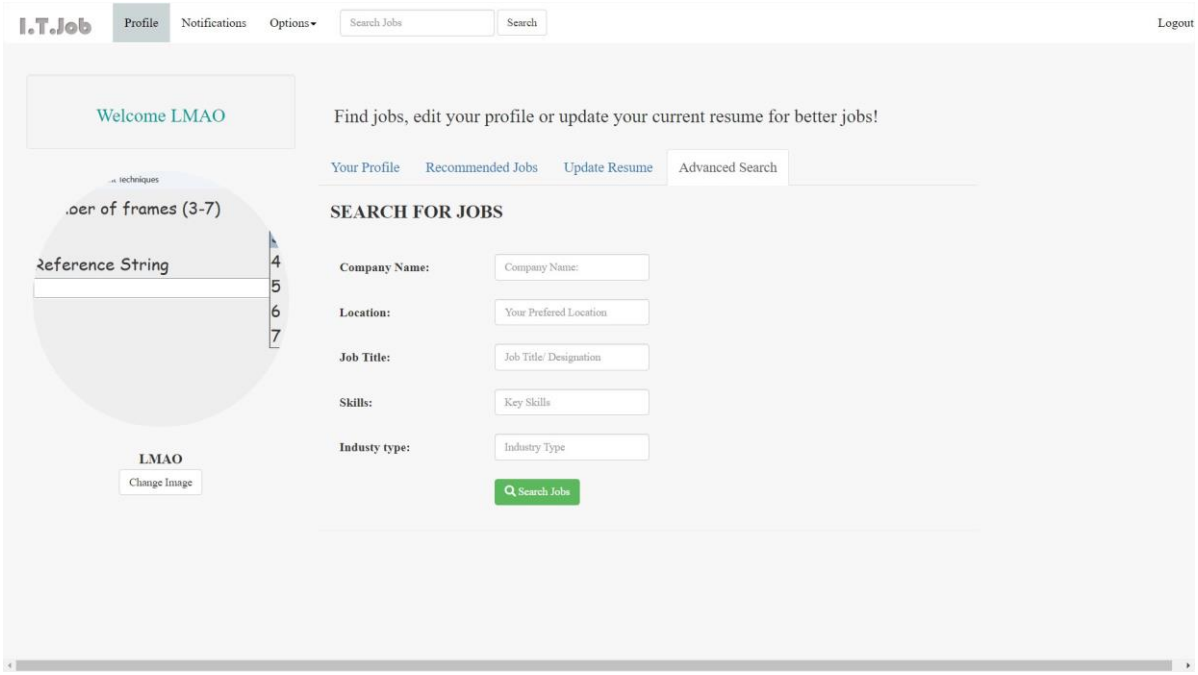
In the main profile, user can view their information as well as choose options for other tasks.



This profile function allow users to see recommended jobs or position based on their provided information. How the recommendation work will be explained in later in this report

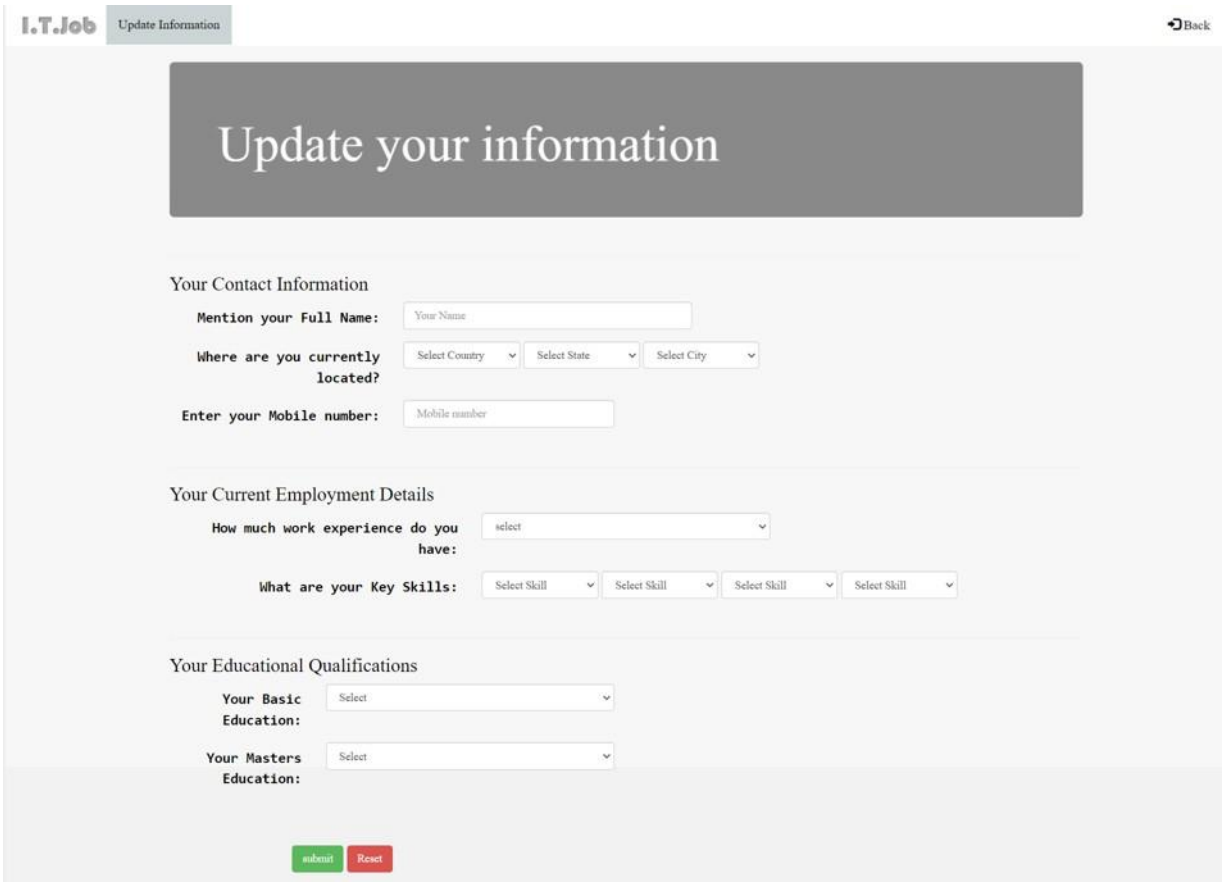


This function enables user to upload or update their CV.

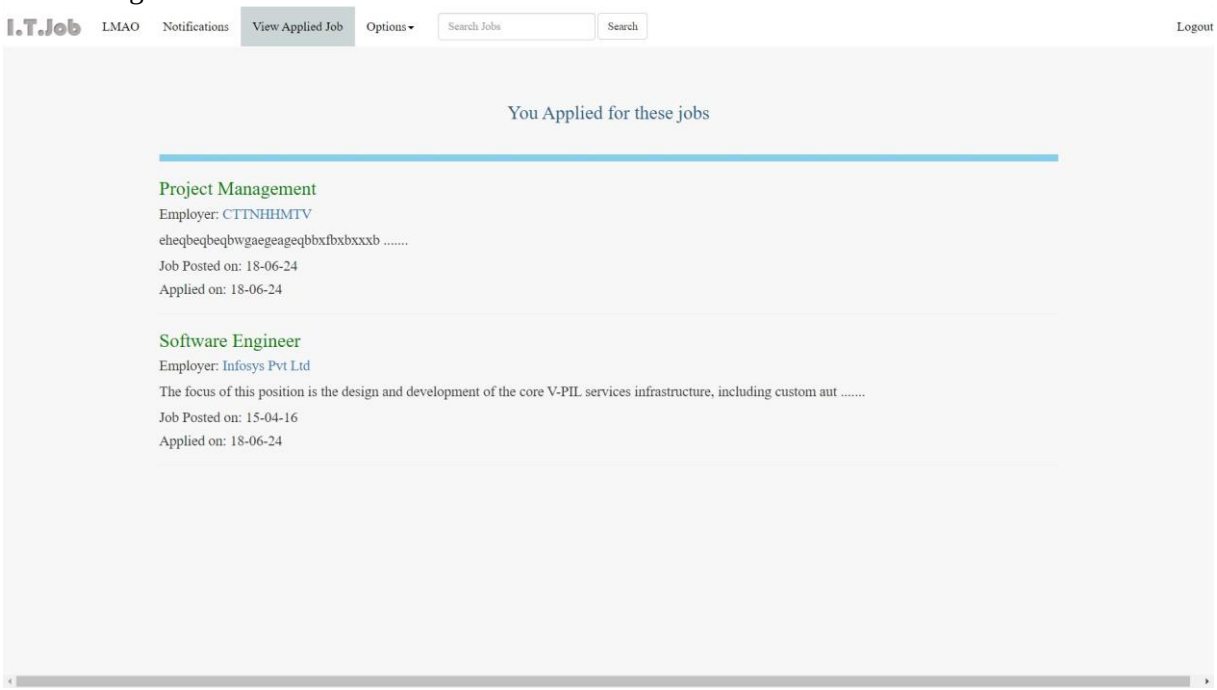


The advance search function, is used to perform an asynchronous HTTP (AJAX) request to a server-side script (adv_search.php) with parameters for company, location, designation, and skills. It dynamically updates the content of an element with the ID subcontent on the webpage without reloading the page. The function handles different states of the request by displaying "Searching.." during the process and showing the result or an error message upon completion.

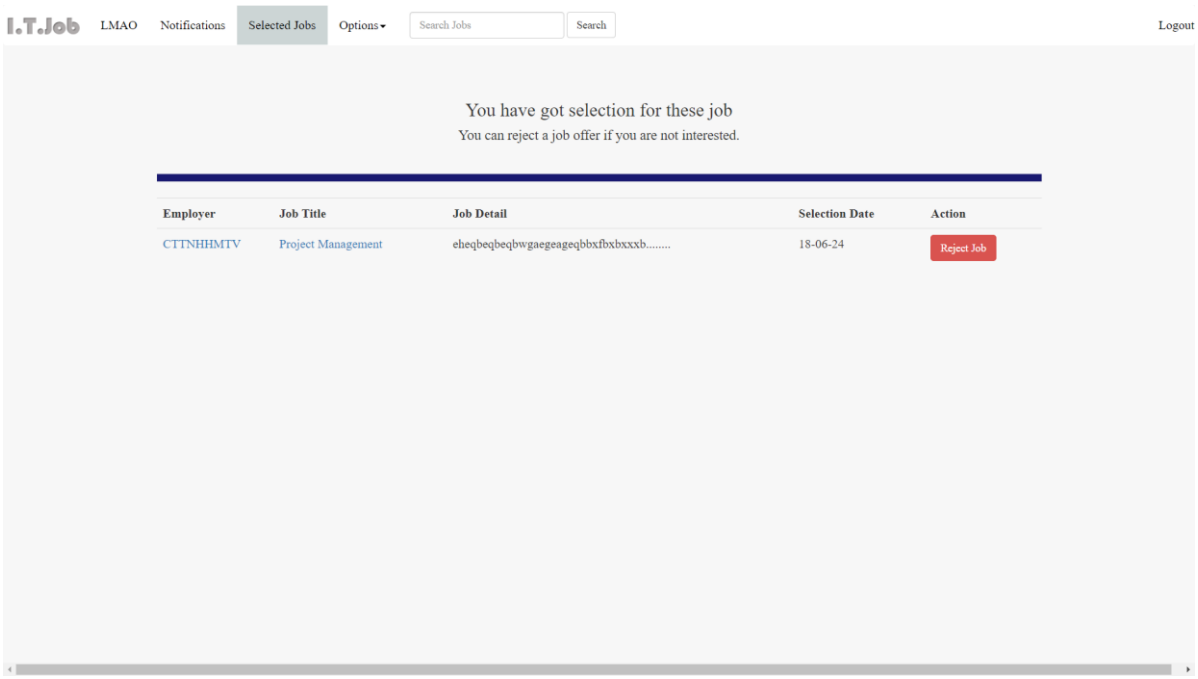
```
function advsearch(){
    var comp=document.getElementById("company").value;
    var location=document.getElementById("location").value;
    ---
}
```



This functions allows user to update their profile by fill in the information the same as the registration form. After that they can choose to proceed to completion, or reset. User also can choose to go back.



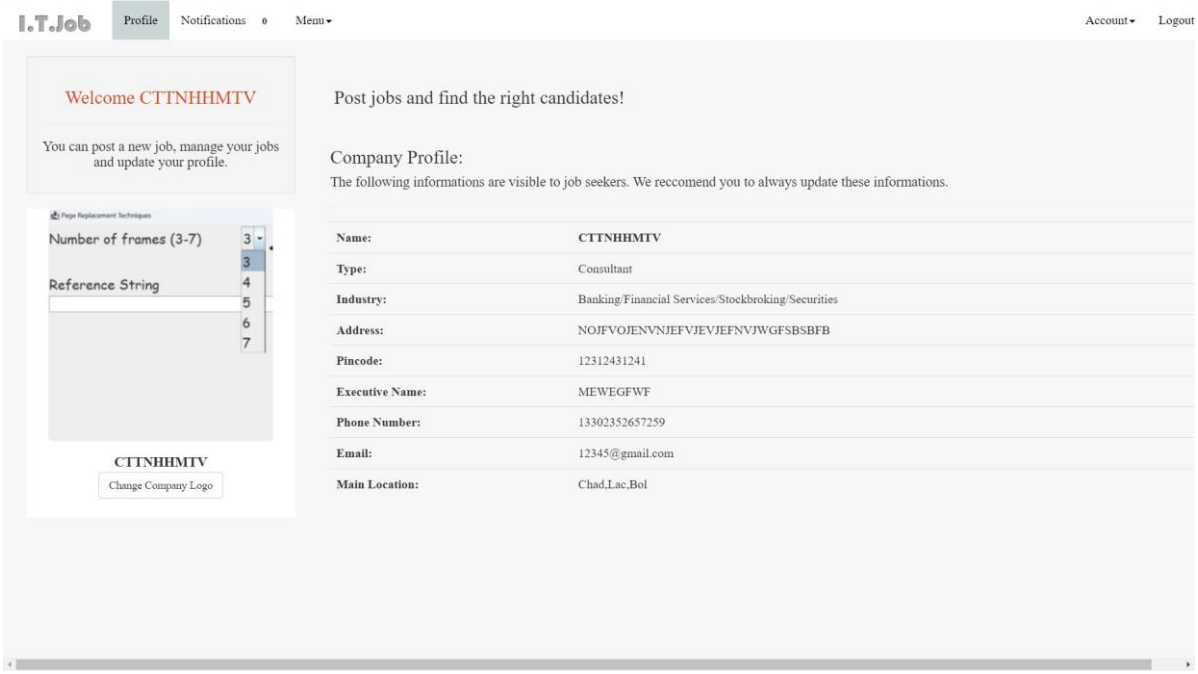
Users can view the list of jobs they applied for along with a brief description, time posted an time applied via this function. And by clicking to a specific job, user can see the job details.



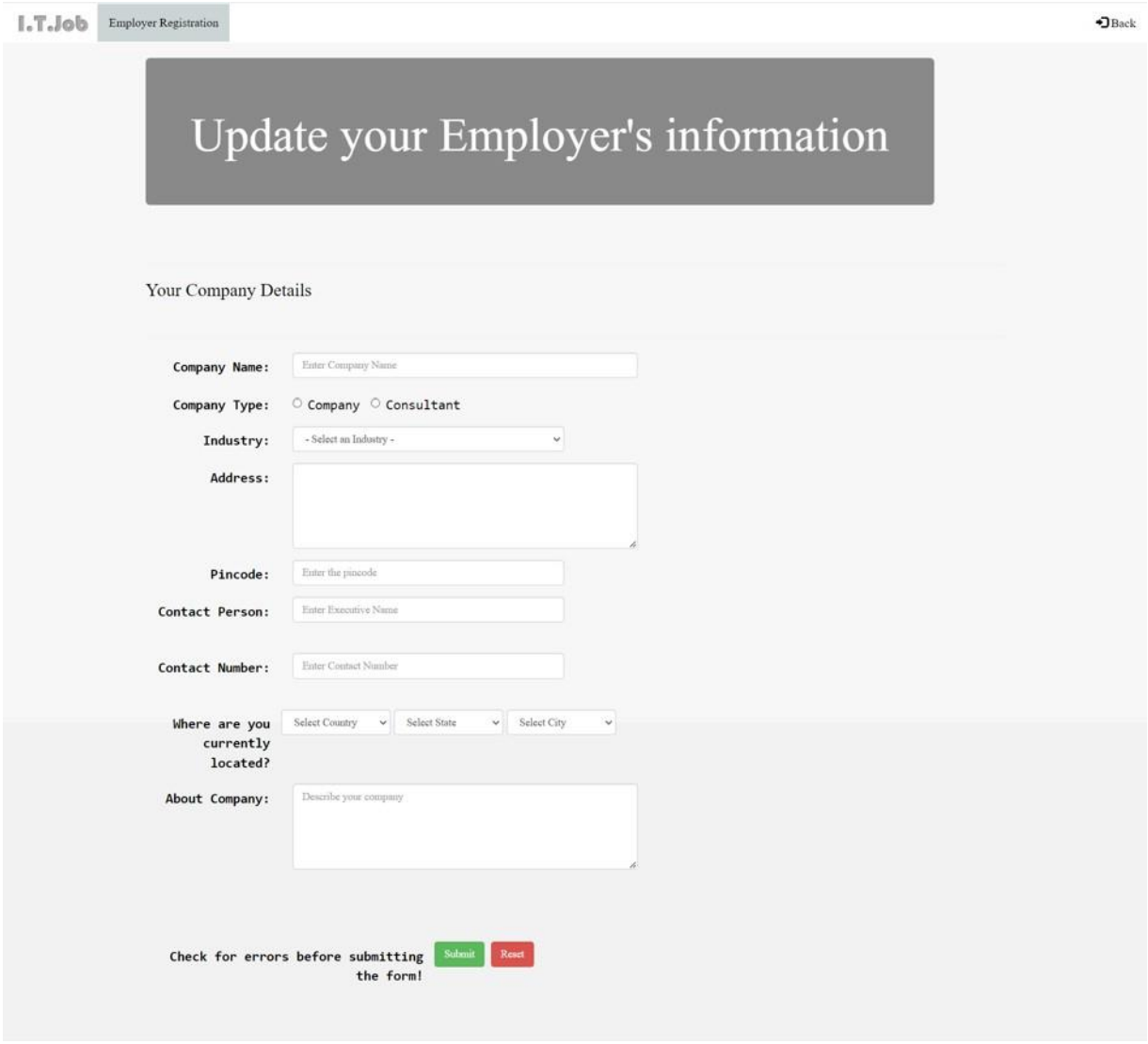
The view selected job function helps users to view a job offer made to them from an employer, user can also choose to reject or accept the offer.

4.6 Employer Profile

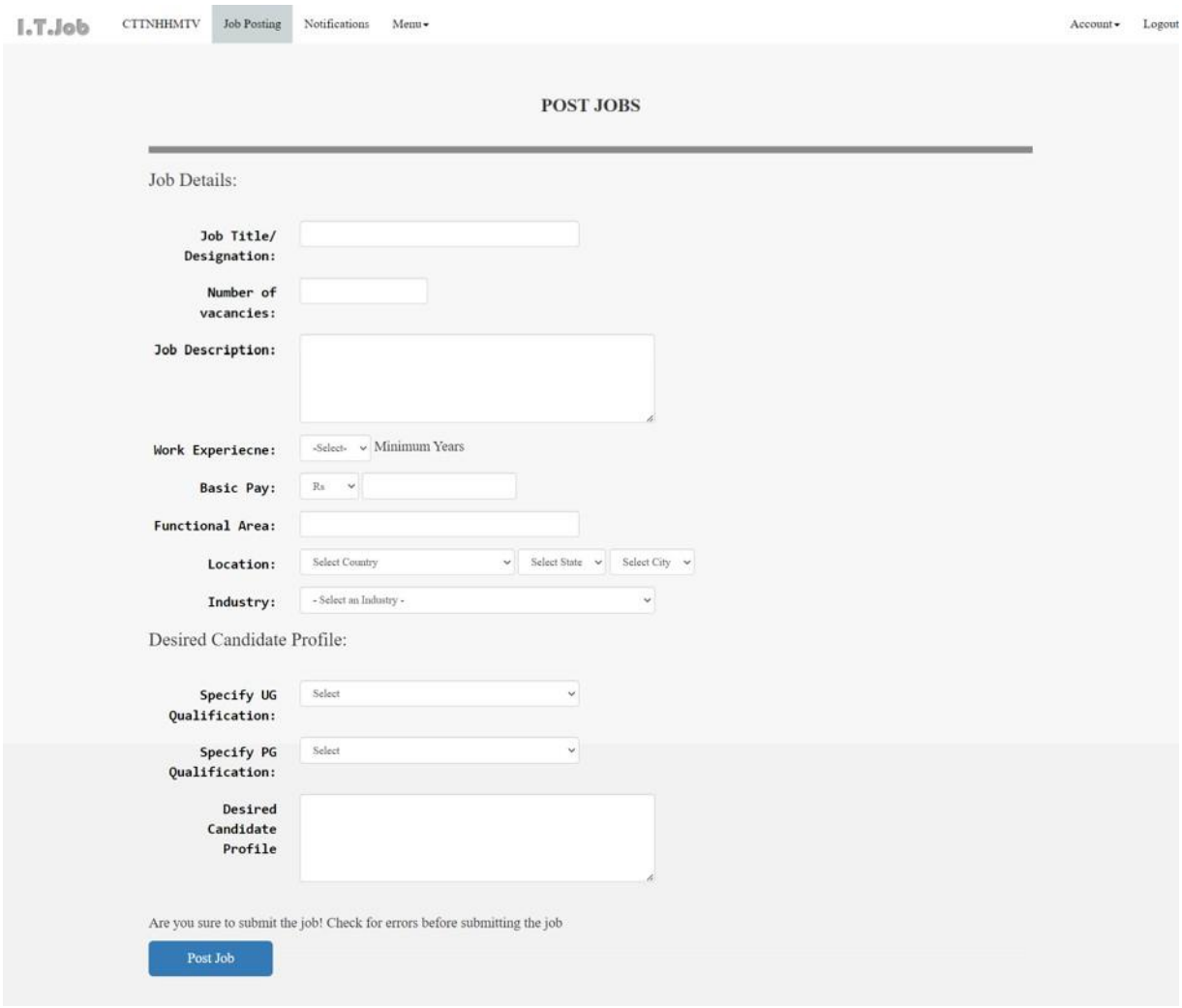
Similar to the jobseeker profile, employer profile also has many functions.



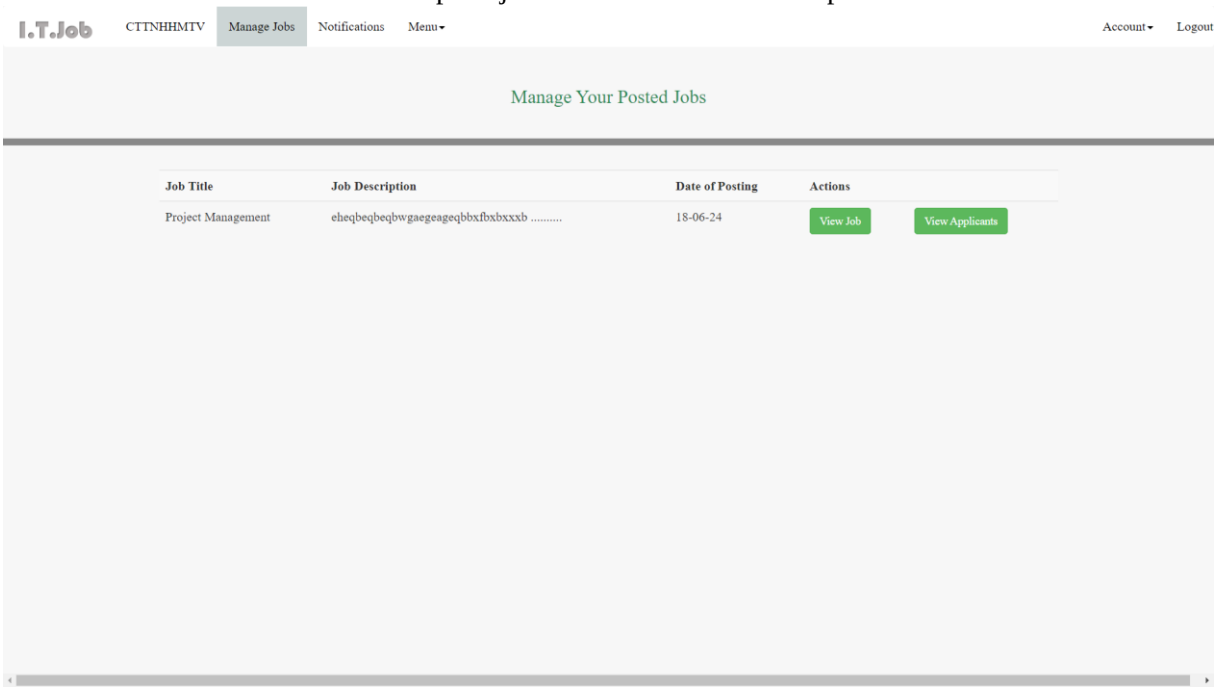
Unlike the jobseeker profile, the employer does not have as many tabs. However, it still allows the user to view their information as well as perform other functions.



This function of employer is close to that of jobseeker, user fill in a form and finish by submitting the updated form. They can also proceed to reset or go back.

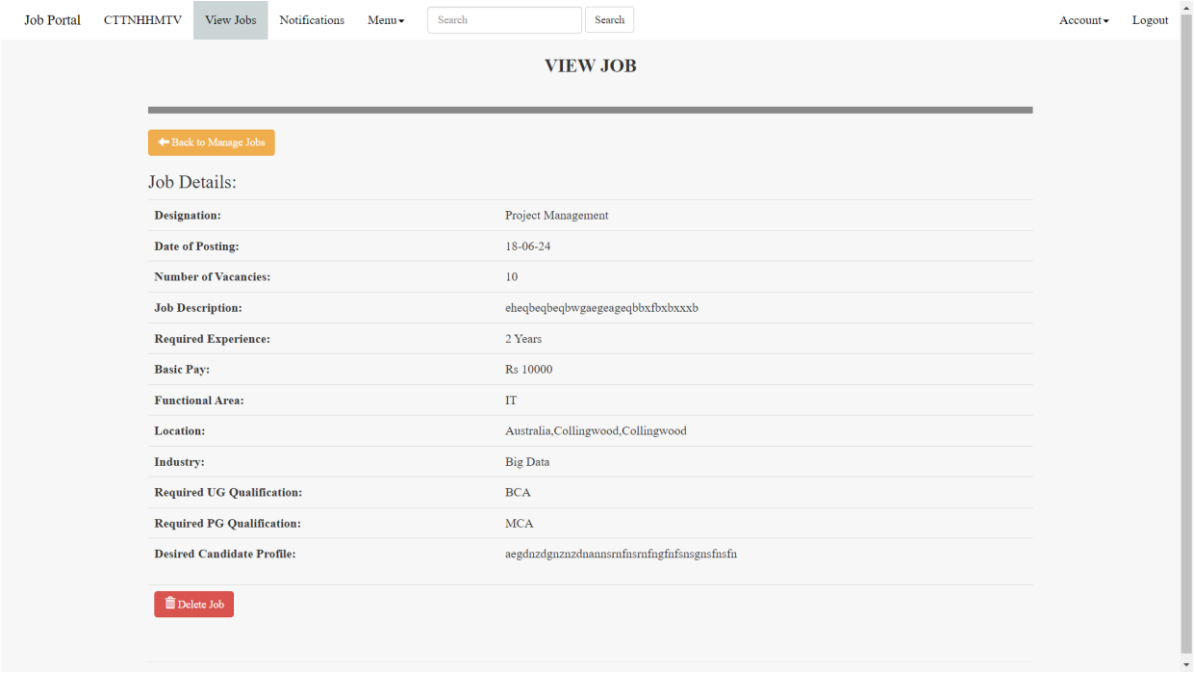


This functions helps employer to post a job by fill in information and requirements in the form of a form. The user then hit the post job button to submit the post.

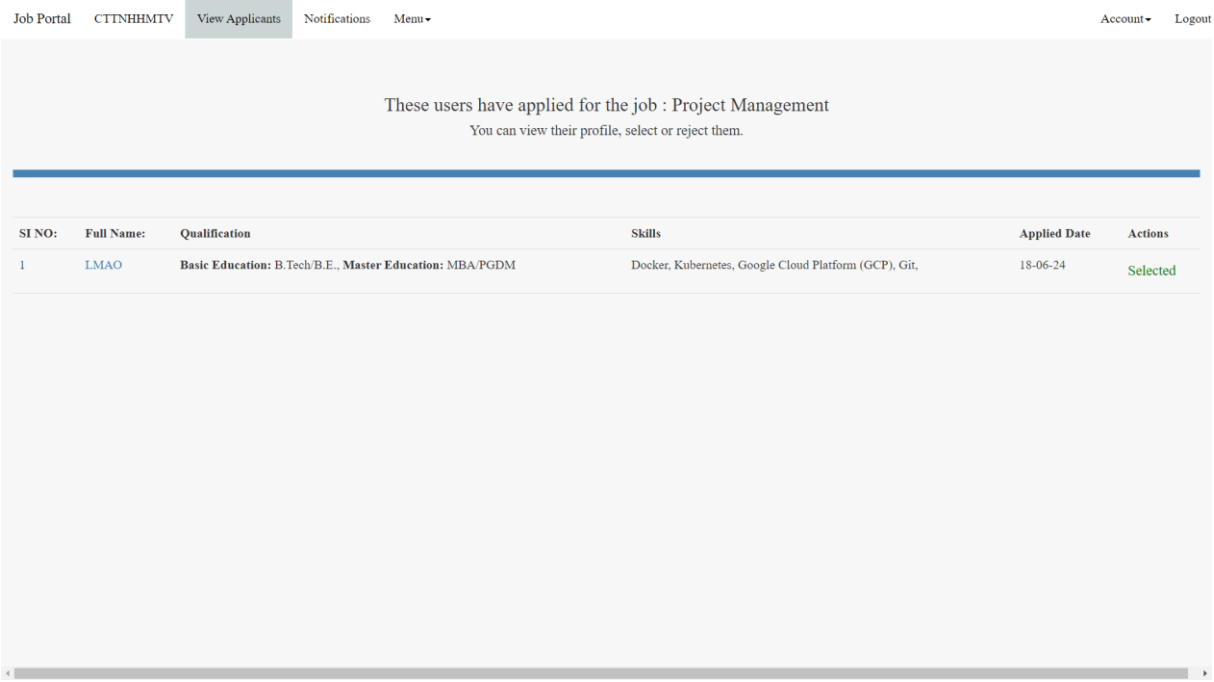


By using this function, user can view the list of jobs with a brief description which they have

already posted. Employer can further proceed by choosing to view job details or view applicants.



The function shows the employer their job's detailed description. It also allow user to delete the posted job or go back to the list.



The employer can see the list of the applicants for their jobs. They can also view the applicants education, qualifications and skills. Employer can either reject or accept the jobseeker

5 Back-end Testing

5.1 Search Jobs

5.1.1 Test case 1: Keyword search successful

```
$keyword= $_GET['key'];
$query = " select * from jobs join employer on jobs. eid = employer. eid where
    title LIKE '%" . $keyword . "%' or employer. ename LIKE '%". $keyword ."%' or
    employer.profile LIKE '%" . $keyword . "%'" ;
$result = mysqli_query($db1, $query);
```

Listing 1: Code for querying database for search function

This code will take the keyword enter by the user and send a query with that keyword to the database to search for the jobs. In this test case the database should return a instances containing the keyword.

Assuming our keyword is Microsoft, we will use the following sql query to query the database:

```
select * from jobs  join employer on jobs.eid = employer.eid  where title LIKE
    'Microsoft' or employer.ename LIKE 'Microsoft' or employer.profile LIKE '
    Microsoft';
```

Listing 2: SQL Code for querying database for search function

Showing rows 0 - 0 (1 total, Query took 0.0008 seconds.)

```
select * from jobportal.jobs join jobportal.employer on jobs.eid = employer.eid where title LIKE
'Microsoft' or employer.ename LIKE 'Microsoft' or employer.profile LIKE 'Microsoft';
```

jobid	eid	title	jobdesc	vacno	experience	basicpay	fnarea	location	industry	ugqual	pgqual
4	2	Web Developer	Development of interactive websites at microfost	5	3	Rs 25000	Web Development	India,Kerala,Ernakulam	Software Services	B.Tech/B.E.	Not Pursuing Post Graduation

5.1.2 Test case 2: Keyword search return null

In this test case the database should return nothing.

Assuming our keyword is google, we will use the following sql query to query the database:

```
select * from jobs  join employer on jobs.eid = employer.eid where title LIKE
    'Google' or employer.ename LIKE 'Google' or employer.profile LIKE 'Google';
```

Listing 3: SQL Code for querying database for search function

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0004 seconds.)

```
select * from jobs join employer on jobs.eid = employer.eid where title LIKE 'Google' or employer.ename
LIKE 'Google' or employer.profile LIKE 'Google';
```

jobid	eid	title	jobdesc	vacno	experience	basicpay	fnarea	location	industry	ugqual	pgqual	profile	postdate	eid	log

Query results operations

Create view

5.2 Register as Employer

5.2.1 Test case 1: Register Successful

```
$query4 = "INSERT INTO login (email,password,usertype,status) VALUES ('$email','
    $hash','employer',0)";

$query5 = "INSERT INTO employer (log_id,ename,phone,location,etype,address,
    pincode,executive,industry)
                VALUES ((SELECT log_id FROM login WHERE email='$email'),'$name
    ','$phone','$location','$type','$addr','$pin','$person','$industry')";
```

Listing 4: Code for querying database for register process

This code inserts a new employer record into the database. In this test case, a new user should be added as an employer to the database.

When a new employer registers on our website by creating an account:

```
INSERT INTO login (email,password,usertype,status) VALUES ('khoinguyen@gmail.com
    ','123','employer',0)";
```

Listing 5: SQL Code for querying database for register process

1 row inserted.
Inserted row id: 41 (Query took 0.0067 seconds.)

```
INSERT INTO login (email,password,usertype,status) VALUES ('khoinguyen@gmail.com','123','employer',0);
```

log_id	email	password	usertype	status
41	khoinguyen@gmail.com	123	employer	0

```
INSERT INTO employer (log_id,ename,phone,location,etype,address,pincode,
    executive,industry)
                VALUES ((SELECT log_id FROM login WHERE email='khoinguyen@gmail.
    com'),'TNHH','0767471717','Vietnam','Company','Vietnam','103123','DaiMinh','
    Steel')";
```

Listing 6: SQL Code for querying database for register process

1 row inserted.
Inserted row id: 7 (Query took 0.0050 seconds.)

```
INSERT INTO employer (log_id,ename,phone,location,etype,address,pincode,executive,industry)VALUES
((SELECT log_id FROM login WHERE
email='khoinguyen@gmail.com'),'TNHH','0767471717','Vietnam','Company','Vietnam','103123','DaiMinh','Ste
el');
```

eid	log_id	ename	etype	industry	address	pincode	executive	phone	location
7	41	TNHH	Company	Steel	Vietnam	103123	DaiMinh	0767471717	Vietnam

5.2.2 Test case 2: Register fail with missing information

In this test case, a new user shouldn't be added as an employer to the database.

```
INSERT INTO login (email,password,usertype,status) VALUES ('khoinguyen@gmail.com','123', ,0);
```

Listing 7: SQL Code for querying database for register process

Error

SQL query: Copy

```
INSERT INTO login (email,password,usertype,status) VALUES ('minhtri@gmail.com','123', ,0);
```

5.3 Register as Jobseeker

5.3.1 Test case 1: Register Successful

```
$query4="INSERT INTO login (email,password,usertype,status) VALUES ('$email','$hash','$type',1)";
    $result1 = mysqli_query($db1,$query4) or die("Cant Register , The user email may be already existing");
$query5 =  "INSERT INTO jobseeker (log_id,name,phone,location,experience,skills,basic_edu,master_edu)
                VALUES ((SELECT log_id FROM login WHERE email='$email'),'$name','$mobile','$location','$experience','$skills','$ug','$pg')";
```

Listing 8: Code for querying database for register process

This code inserts a new jobseeker record into the database. In this test case, a new user should be added as an jobseeker to the database.

When a new jobseeker registers on our website by creating an account:

```
INSERT INTO login (email,password,usertype,status) VALUES ('minhtri@gmail.com','123','jobseeker',1);
```

Listing 9: SQL Code for querying database for register process

```
INSERT INTO jobseeker (log_id,name,phone,location,experience,skills,basic_edu,master_edu) VALUES ((SELECT log_id FROM login WHERE email='minhtri@gmail.com'),'Minh Tri','0767470100','Vietnam','3 Years','C++','BA','CS');
```

Listing 10: SQL Code for querying database for register process

1 row inserted.
Inserted row id: 44 (Query took 0.0056 seconds.)

```
INSERT INTO login (email,password,usertype,status) VALUES ('Daiminh@gmail.com','123','CEO',1);
```

1 row inserted.
Inserted row id: 20 (Query took 0.0060 seconds.)

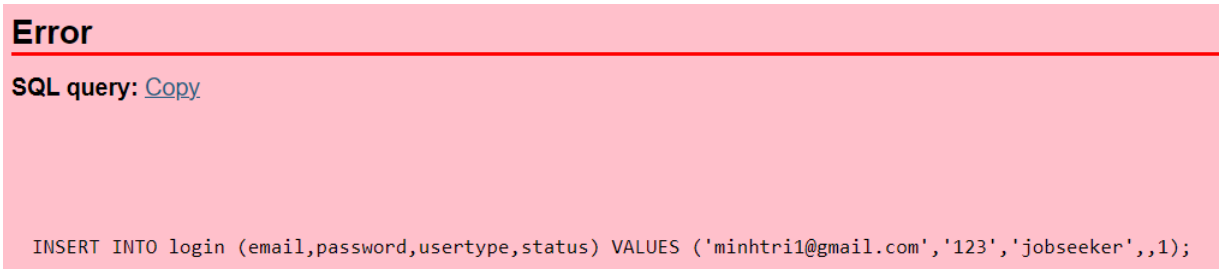
```
INSERT INTO jobseeker (log_id,name,phone,location,experience,skills,basic_edu,master_edu) VALUES ((SELECT log_id FROM login WHERE email='Daiminh@gmail.com'),'Dai Minh','0767879198','Vietnam','3 Years','C++','B.A','CS');
```

5.3.2 Test case 2: Register fail with missing information

In this test case, a new user shouldn't be added as an employer to the database.

```
INSERT INTO login (email,password,usertype,status) VALUES ('minhtri1@gmail.com',
    '123','jobseeker',,1);
```

Listing 11: SQL Code for querying database for register process



5.4 Edit profile

5.4.1 Test case 1: Edit Profile Jobseeker

```
    // Prepare an UPDATE statement
    $sql = "UPDATE jobseeker SET name=?, location=?, phone=?, experience=?,
    skills=?, basic_edu=?, master_edu=? WHERE user_id=?";
    if ($stmt = mysqli_prepare($db1, $sql)) {
        // Bind variables to the prepared statement as parameters
        mysqli_stmt_bind_param($stmt, "sssisssi", $name, $location, $phone,
    $experience, $skills, $basic_edu, $master_edu, $_SESSION['jsid']);

        // Attempt to execute the prepared statement
        if (mysqli_stmt_execute($stmt)) {
            // Redirect to profile page
            header("location: profile.php");
            exit();
        } else {
            echo "Something went wrong. Please try again later.";
        }
    }
```

Listing 12: Code for querying database for edit process

This code update a new jobseeker record into the database. In this test case, information of old user should be updated.

```
UPDATE jobseeker SET name='Minh Dai', location='Vietnam', phone='0976854344',
    experience='3', skills='C++', basic_edu='B.A' WHERE user_id=20;
```

Listing 13: SQL Code for querying database for edit process

```
1 row affected. (Query took 0.0076 seconds.)
UPDATE jobseeker SET name='Minh Dai', location='Vietnam', phone='0976854344', experience='3', skills='C++', basic_edu='B.A' WHERE
user_id=20;
```

5.4.2 Test case 2: Edit Profile Employer

```
1     // Prepare an UPDATE statement
2     $sql = "UPDATE employer SET ename=?, etype=?, industry=?, address=?, pincode
      =?, executive=?, phone=?, location=? WHERE eid=?";
3
4
5     if ($stmt = mysqli_prepare($db1, $sql)) {
6         // Bind variables to the prepared statement as parameters
7         mysqli_stmt_bind_param($stmt, "ssssisssi", $name, $type, $industry,
      $addr, $pin, $person, $phone, $location, $_SESSION['eid']);
8
9         // Attempt to execute the prepared statement
10        if (mysqli_stmt_execute($stmt)) {
11            // Redirect to profile page
12            header("location: profile.php");
13            exit();
14        } else {
15            echo "Something went wrong. Please try again later.";
16        }
17    }
```

Listing 14: Code for querying database for edit process

This code update a new employer record into the database. In this test case, information of old user should also be updated.

```
1 UPDATE employer SET ename='LMFAO', etype='asdfasdf', industry='asdfasdf',
    address='adsasdf', pincode='1231', executive='MAO', phone='0978586456',
    location='Vietnam' WHERE eid=5;
```

Listing 15: SQL Code for querying database for edit process

```
1 row affected. (Query took 0.0048 seconds.)
UPDATE employer SET ename='LMFAO', etype='asdfasdf', industry='asdfasdf', address='adsasdf', pincode='1231', executive='MAO',
phone='0978586456', location='Vietnam' WHERE eid=5;
```

5.5 Job application

5.5.1 Test case 1: Job application is already in

```
1 $q1=mysqli_query($db1,"select * from application where job_id =$jobid AND
    user_id = $jsid ");
2 if( mysqli_num_rows ( $q1 ) >=1){
3     echo " <div class='alert alert-danger alert-dismissible' role='alert'>
```

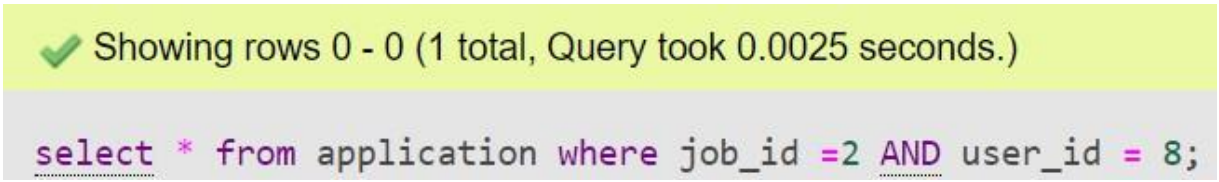
```
            <button type='button' class='close'  data-dismiss='alert' aria-label
    ='Close'><span
                    aria-hidden='true'>&times;</span></button>
          <p style='font-size: 20px'><strong>Note:</strong> You have already
    applied for this job!</p>
        </div>";
}
```

Listing 16: Code for querying database for application process

This code will test job application. In this case, there is already a job application so the query should return a result and the frontend will display the alert.

```
select * from application where job_id =2 AND user_id = 8;
```

Listing 17: SQL Code for querying database for application process



5.5.2 Test case 2: Job application is new

```
else{
    $q2=mysqli_query($db1,"insert into application (user_id,emp_id,job_id,
    date_applied) VALUES ($jsid,(SELECT eid from jobs where jobid = $jobid),
    $jobid,'$date')");
    if($q2){
        echo " <div class='alert alert-success alert-dismissible' role='alert'>
            <button type='button' class='close'  data-dismiss='alert' aria-label
    ='Close'><span
                    aria-hidden='true'>&times;</span></button>
          <p style='font-size: 20px'><strong>Note:</strong> You have
    successfully applied for this job!</p>
        </div>";
    }
```

Listing 18: Code for querying database for application process

This code will insert into the database application the user and will notify about the successful application.

```
insert into application (user_id,emp_id,job_id,date_applied) VALUES (10,(SELECT
    eid from jobs where jobid = 2),2,'18-04-16');
```

Listing 19: SQL Code for querying database for register process

5.6 Post jobs

5.6.1 Test case 1: Post Jobs successful

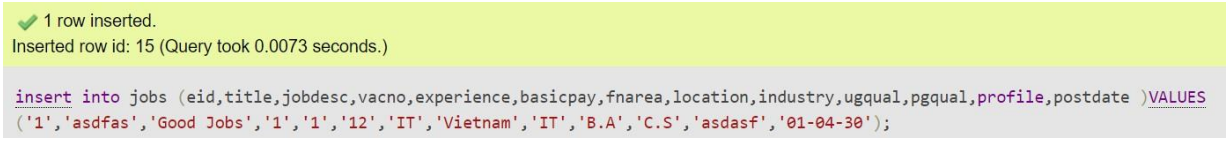
```
$query4="insert into jobs (eid,title,jobdesc,vacno,experience,basicpay,fnarea,
    location,industry,ugqual,pgqual,profile,postdate )VALUES ('$eid','$desig','
    $desc','$vacno','$exp','$pay','$fnarea','$location','$indtype','$ug','$pg','
    $profile','$date')";

if (!mysqli_query($db1,$query4))
{
 echo("Error description: " . mysqli_error($db1));
}
```

Listing 20: Code for querying database for post jobs process

This code will insert the job listing into the database. In this case, a successful insert will appear.

```
insert into jobs (eid,title,jobdesc,vacno,experience,basicpay,fnarea,location,
    industry,ugqual,pgqual,profile,postdate )VALUES ('1','asdfas','Good Jobs','1
    ','1','12','IT','Vietnam','IT','B.A','C.S','asdasf','01-04-30');
```

Listing 21: SQL Code for querying database for post jobs process



5.6.2 Test case 2: Post Jobs unsuccessful

In this test case, the query will have some parameter missing so it should return error.

```
insert into jobs (eid,title,jobdesc,vacno,experience,basicpay,fnarea,location,
    industry,ugqual,pgqual,profile,postdate )VALUES ('1','asdfas','Good Jobs','1
    ','1','12','IT','Vietnam','IT','B.A','C.S','asdasf',);
```

Listing 22: SQL Code for querying database for post jobs process

5.7 Delete jobs

5.7.1 Test case 1: Delete successful

```
$query = "DELETE FROM jobs WHERE jobid=$_GET[jid]";
$result = mysqli_query($db1,$query);
if( $result) {
    echo "<h3 style='color: green;'> Selected Job Is Successfully Deleted</h3>";
}
```

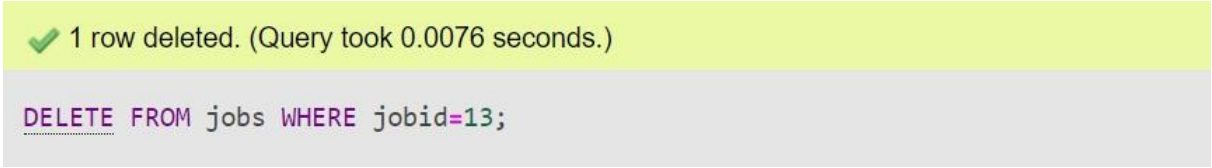
```
else{
    echo "<h3 style='color: red;'> Failed to delete the selected job!</h3>";
}
```

Listing 23: Code for querying database for delete process

This code will delete the job in the database. In this case, a successful delete will appear

```
DELETE FROM jobs WHERE jobid=13;
```

Listing 24: SQL Code for querying database for delete process

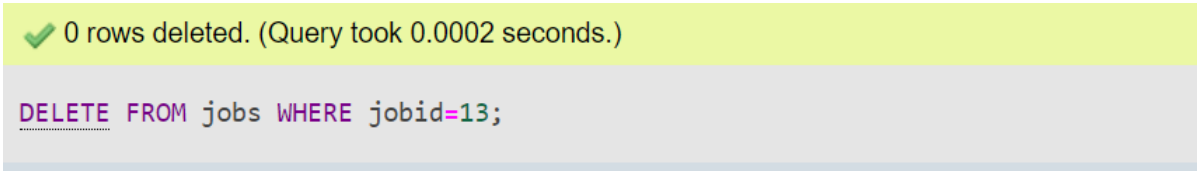


5.7.2 Test case 2: Delete fail

In this case, a failure delete will show 0 rows deleted

```
DELETE FROM jobs WHERE jobid=13;
```

Listing 25: SQL Code for querying database for delete process



6 Front-end Testing

6.1 Main Page

These are the functions of the Main Page.

URL: http://localhost/job_portal/login.php

These are the commands:

✓ mainPage

	Command	Target	Value
1	✓ open	/job_portal/	
2	✓ set window size	652x720	
3	✓ click	id=keyword	
4	✓ type	id=keyword	Micro
5	✓ click	css=.glyphicon-search	
6	✓ mouse over	css=.glyphicon-search	
7	✓ mouse out	css=.glyphicon-search	
8	✓ click	css=.glyphicon-chevron-up	
9	✓ click	css=.col-sm-6:nth-child(1) .btn	
10	✓ run script	window.scrollTo(0,713.3333129882812)	
11	✓ click	css=.col-sm-6:nth-child(2) .btn	
12	✓ click	css=.img-fluid	

✓ mainPage

	Command	Target	Value
12	✓ click	css=.img-fluid	
13	✓ click	linkText=Register	
14	✓ click	linkText=Become a Jobseeker	
15	✓ click	css=.img-fluid	
16	✓ click	linkText=Register	
17	✓ click	linkText=Become an Employer	
18	✓ click	css=.img-fluid	
19	✓ click	linkText=Login	
20	✓ mouse over	linkText=Sign Up	
21	✓ mouse out	linkText=Sign Up	
22	✓ click	css=.img-fluid	

6.2 Login Page

These are the functions of the Login Page.

URL: http://localhost/job_portal/login.php

These are the commands:

✓ EmpRegistration*
✓ JobseekerRegistration*
✓ LoginPage*
✓ mainPage

	Command	Target	Value
1	✓ open	http://localhost/job_portal/login.php	
2	✓ set window size	652x726	
3	✓ click	linkText=Register Today	
4	✓ click	linkText=Login	
5	✓ click	css=.col-lg-4:nth-child(1) .btn	
6	✓ click	linkText=Login	
7	✓ click	id=inputEmail	
8	✓ type	id=inputEmail	12345@gmail.com
9	✓ type	id=inputPassword	12345678
10	✓ click	css=.btn-lg	

6.3 Jobseeker Registration Page

These are the functions of the Jobseeker Registration Page.

URL: http://localhost/job_portal/login.php

These are the commands:

✓ JobseekerRegistration*
✓ mainPage

	Command	Target	Value
1	✓ open	http://localhost/job_portal/jobseeker/register_user.php	
2	✓ set window size	652x726	
3	✓ type	id=email	Nguyen@gmail.com
4	✓ type	id=passnew	12345678
5	✓ type	id=passconf	12345678
6	✓ click	id=name	
7	✓ click	id=name	
8	✓ type	id=name	Nguyen Khoi Nguyen
9	✓ click	id=countryId	
10	✓ select	id=countryId	label=Vietnam
11	✓ click	id=stateId	
12	✓ select	id=stateId	label=Mien Nui Va Trung Du

✓ JobseekerRegistration*
✓ mainPage

	Command	Target	Value
13	✓ click	id=cityId	
14	✓ click	id=mobno	
15	✓ type	id=mobno	0767470100
16	✓ click	id=experience	
17	✓ select	id=experience	label=2 year
18	✓ click	id=skill1	
19	✓ select	id=skill1	label=C++
20	✓ click	id=skill2	
21	✓ select	id=skill2	label=SQL
22	✓ click	id=skill3	
23	✓ select	id=skill3	label=Ruby
24	✓ click	id=skill4	

✓ JobseekerRegistration*
✓ mainPage

	Command	Target	Value
20	✓ click	id=skill2	
21	✓ select	id=skill2	label=SQL
22	✓ click	id=skill3	
23	✓ select	id=skill3	label=Ruby
24	✓ click	id=skill4	
25	✓ select	id=skill4	label=HTML/CSS
26	✓ click	id=ugcourse	
27	✓ select	id=ugcourse	label=B.A
28	✓ click	id=pgcourse	
29	✓ select	id=pgcourse	label=CS
30	✓ click	id=reg	

6.4 Employer Registration Page

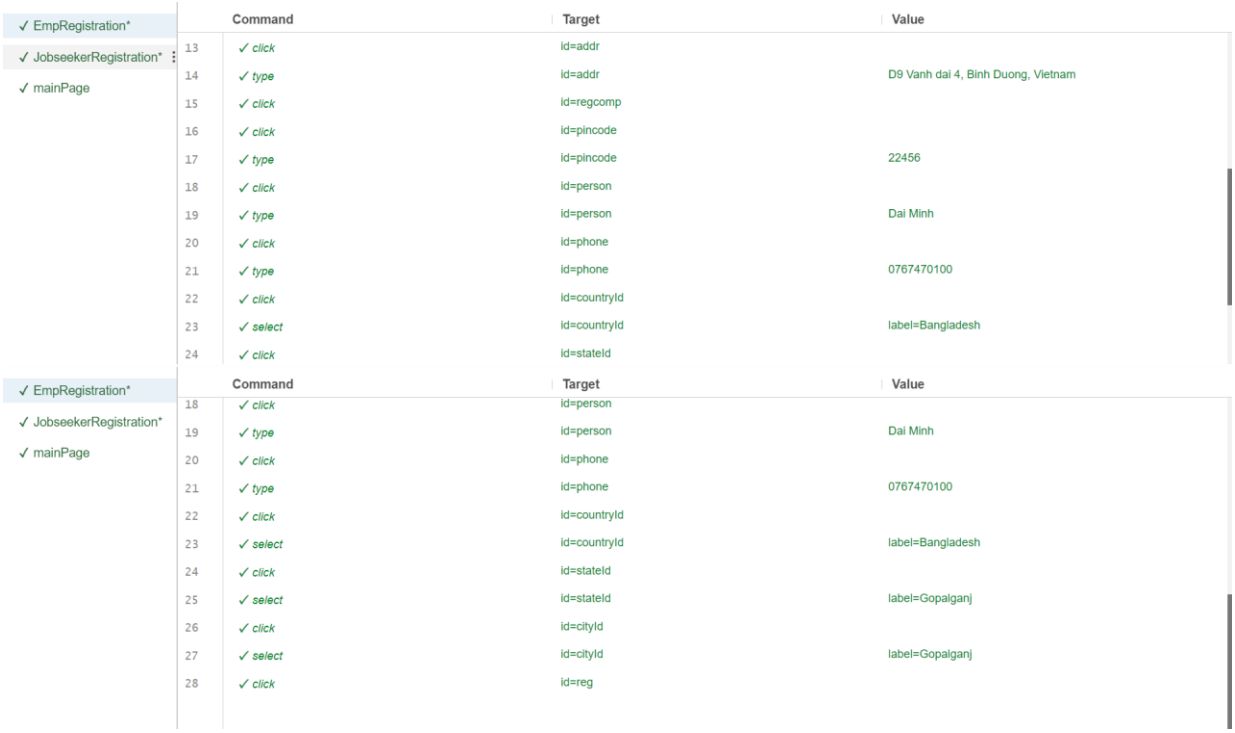
These are the functions of the Employer Registration Page.

URL: http://localhost/job_portal/login.php

These are the commands:

✓ EmpRegistration*
✓ JobseekerRegistration*
✓ mainPage

	Command	Target	Value
1	✓ open	http://localhost/job_portal/employer/register_emp.php	
2	✓ set window size	652x720	
3	✓ click	id=email	
4	✓ type	id=email	Nghi@gmail.com
5	✓ type	id=pass1	12345678
6	✓ type	id=pass2	12345678
7	✓ type	id=compname	KhanhNghiTNHH
8	✓ click	css=.radio-inline:nth-child(1)	
9	✓ click	id=indtype	
10	✓ select	id=indtype	label=Brewery/Distillery
11	✓ click	id=indtype	
12	✓ click	css=.form-group:nth-child(10)	

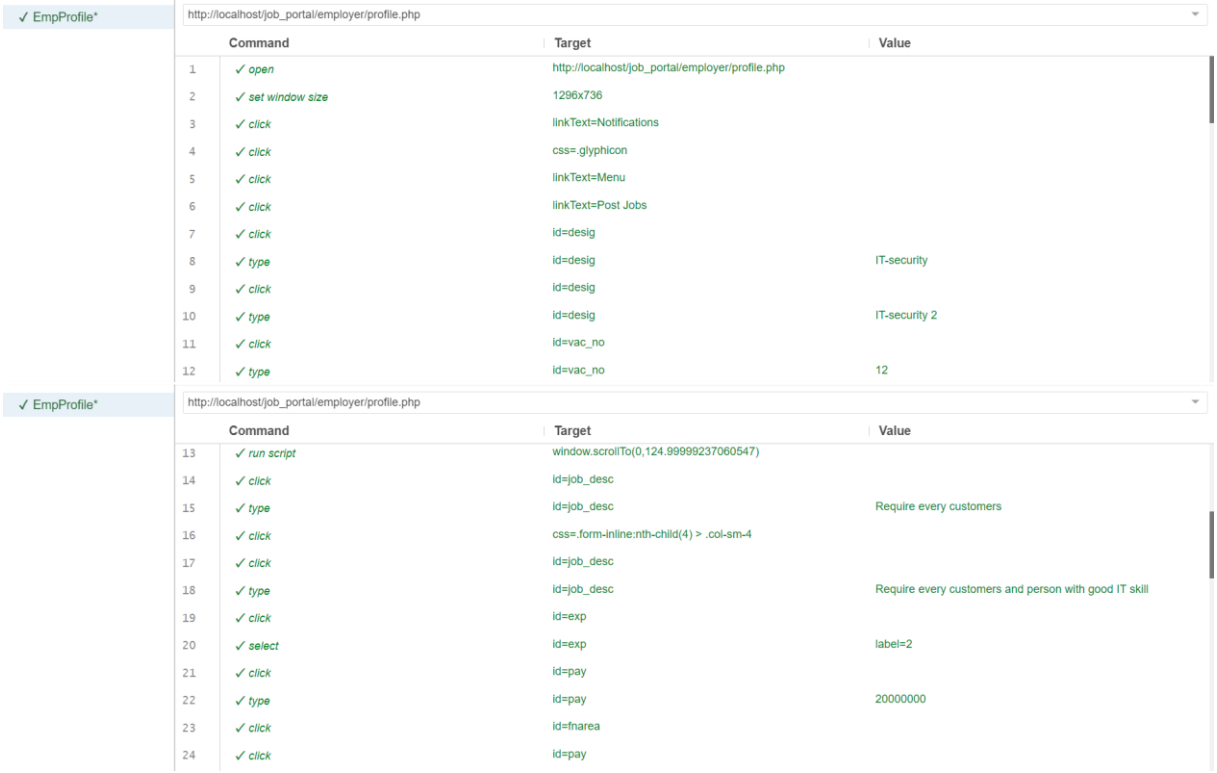


6.5 Employer Page

These are the functions of the Employer Page.

URL: http://localhost/job_portal/login.php

These are the commands:



✓ EmpProfile*

http://localhost/job_portal/employer/profile.php

	Command	Target	Value
25	✓ click	id=pay	
26	✓ double click	id=pay	
27	✓ type	id=pay	200000
28	✓ send keys	id=pay	${KEY_ENTER}
29	✓ click	id=fnarea	
30	✓ type	id=fnarea	Distric 7
31	✓ click	id=countryId	
32	✓ select	id=countryId	label=Saint Helena
33	✓ click	id=stateId	
34	✓ select	id=stateId	label=Ascension
35	✓ click	id=cityId	
36	✓ select	id=cityId	label=Georgetown

✓ EmpProfile*

http://localhost/job_portal/employer/profile.php

	Command	Target	Value
37	✓ click	id=indtype	
38	✓ select	id=indtype	label=Quality Assurance (QA) and Testing
39	✓ click	id=ugcourse	
40	✓ select	id=ugcourse	label=B.A
41	✓ click	id=pgcourse	
42	✓ select	id=pgcourse	label=CS
43	✓ click	id=profile	
44	✓ click	css=#job_post > .form-group:nth-child(3)	
45	✓ click	id=profile	
46	✓ click	id=profile	
47	✓ type	id=profile	Require every customers and person with good IT skill Requ ire every customers and person with good IT skill

✓ EmpProfile*

http://localhost/job_portal/employer/profile.php

	Command	Target	Value
47	✓ type	id=profile	Require every customers and person with good IT skill Requ ire every customers and person with good IT skill
48	✓ mouse over	id=postbtn	
49	✓ click	id=postbtn	
50	✓ click	linkText=Menu	
51	✓ click	linkText=Manage Jobs	
52	✓ click	css=td:nth-child(4) .btn	
53	✓ click	css=.btn-warning	
54	✓ click	css=td:nth-child(5) .btn	
55	✓ click	css=.nav:nth-child(2) > li:nth-child(1) > a	
56	✓ mouse over	css=.active > a	
57	✓ mouse out	css=.active > a	

6.6 Jobseeker Page

These are the functions of the Jobseeker Page.

URL: http://localhost/job_portal/login.php

These are the commands:

✓ EmpProfile*
✓ EmpRegistration
✓ JobSeekerProfile*
✓ JobseekerRegistration
✓ LoginPage
✓ mainPage

	Command	Target	Value
1	✓ open	http://localhost/job_portal/jobseeker/profile.php	
2	✓ set window size	1296x736	
3	✓ click	linkText=Recommended Jobs	
4	✓ click	linkText=Update Resume	
5	✓ click	linkText=Advanced Search	
6	✓ click	linkText=Your Profile	
7	✓ click	id=keyword	
8	✓ type	id=keyword	a
9	✓ click	css=.btn-default:nth-child(2)	
10	✓ click	linkText=LMFAO	
11	✓ click	css=.nav:nth-child(2) > li:nth-child(1) > a	
12	✓ click	linkText=Notifications	

7 Security

In today's digital age, user account security is paramount, and one common method to ensure this security is by storing user accounts in a local database with hashed passwords. This approach has several advantages and disadvantages, which are crucial for organizations to consider when designing their systems.

Pros:

1. Enhanced Security: Storing passwords in a hashed format significantly enhances security. Hashing algorithms convert plain-text passwords into fixed-size strings of characters, which are irreversible. This means that even if an attacker gains access to the database, they cannot easily retrieve the original passwords. Using strong, modern hashing algorithms like bcrypt or Argon2 adds layers of security, making it extremely difficult for attackers to crack the hashed passwords.

2. Control and Customization: A local database offers full control over the data and how it is managed. Organizations can implement custom security policies, such as specific hashing algorithms, salting (adding random data to passwords before hashing), and periodic rehashing to enhance security. This control also extends to access permissions, backup procedures, and data recovery processes tailored to the organization's specific needs.

3. Performance: Storing user data locally can lead to improved performance. Local databases typically offer faster data retrieval and response times compared to remote servers, especially for applications with high traffic or requiring quick authentication processes. This speed can enhance user experience and reduce latency issues.

4. Cost-Effective: Utilizing a local database can be cost-effective for small to medium-sized organizations. It eliminates the need for expensive third-party services and allows for more predictable budgeting. The costs associated with maintaining local servers and databases can be lower compared to subscription fees for cloud services, especially if the required infrastructure is already in place.

Cons:

1. Maintenance and Management: Managing a local database requires ongoing maintenance, including software updates, security patches, and hardware upkeep. Organizations must have dedicated IT staff to handle these tasks, which can be resource-intensive and costly. Failure to maintain the database properly can lead to vulnerabilities and potential data breaches.

2. Scalability Issues: As the organization grows, so does the amount of data. Scaling a local database to accommodate increasing user accounts and data can be challenging. It often requires additional hardware, storage, and network resources, which can be expensive and logistically complex. Cloud-based solutions, in contrast, offer more seamless scalability options.

3. Security Risks: While hashed passwords provide a layer of security, local databases are still susceptible to physical and cyber threats. Physical threats include theft or damage to the server hardware, while cyber threats involve hacking attempts, malware, and other forms of cyberattacks. Comprehensive security measures, such as firewalls, encryption, and intrusion detection systems, are essential but add to the complexity and cost of managing local databases.

Password Recovery:

If there is a situation where users forgot their password it may be recover by authentication via the provided email of the user. This might involve working with the database and a library where SMTP connection is allowed and email connection is available. An email containing a code to recover password will be available and once user receive it they can recover and update their passwords.

Access Control:

There are also certain components only available to certain users.

1. Jobseeker: Jobseeker will have access to their own user information as well as the management of their own profile page and application.

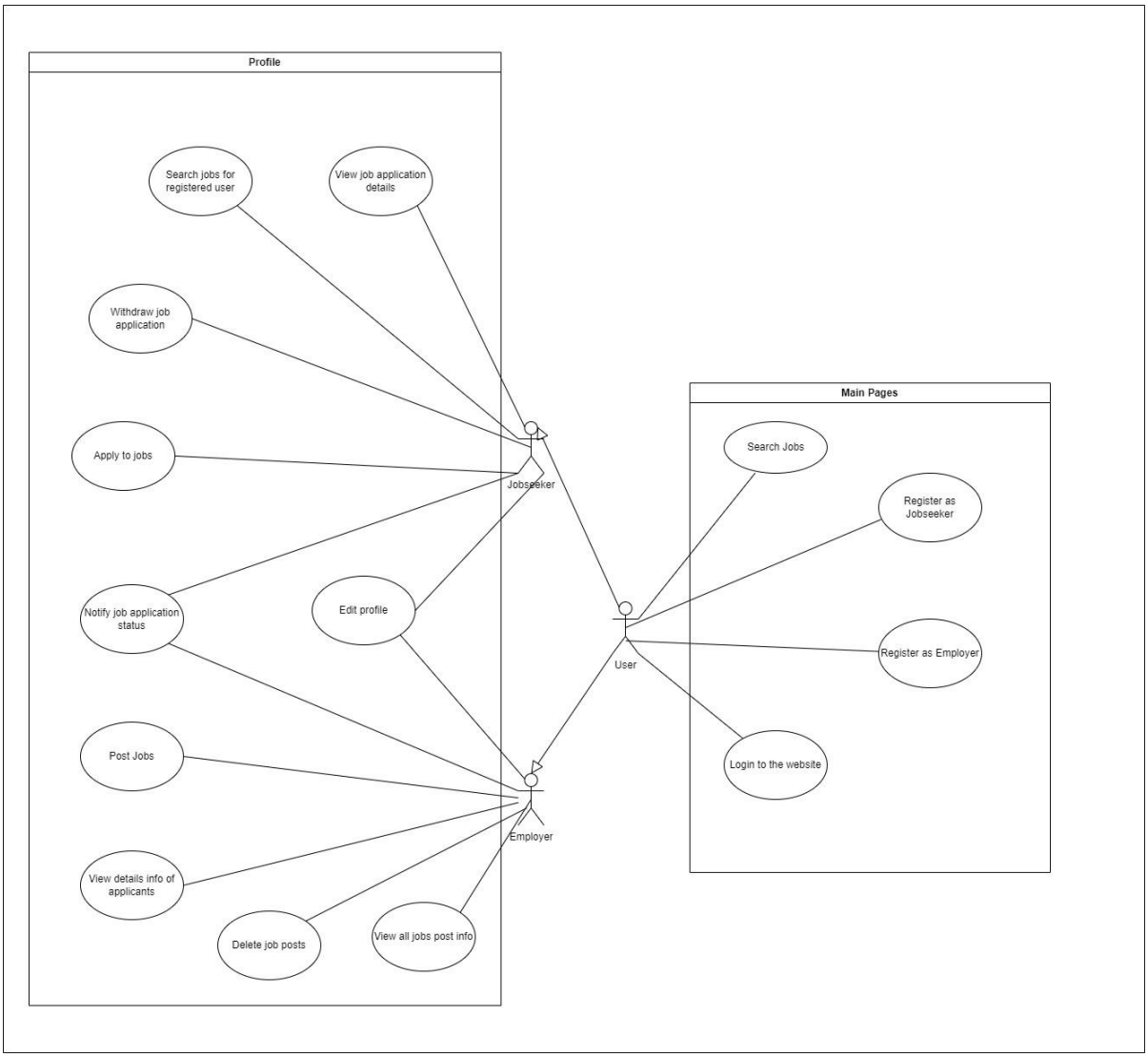
2. Employer: Employer will have control over jobs management abilities such as post jobs, delete jobs,... as well as having management over their applicants for example reject or accept and applicants.

3. Non-register User: Users who have not registered is only allowed to view available jobs and search for them but can not look into any further information unless they register.

8 Change Management

On June 11th, the customers have reviewed our project progression thoroughly and we have received request to add in a notification system for both the employer as well as the jobseeker. We have decided to update 1 more use case to our use case diagrams, do a thorough use case analysis of the use case as well as create a new sequence diagram for such use case. Aside from that we also slightly modify our mockup UI to incorporate the use case and we also updated our code to match.

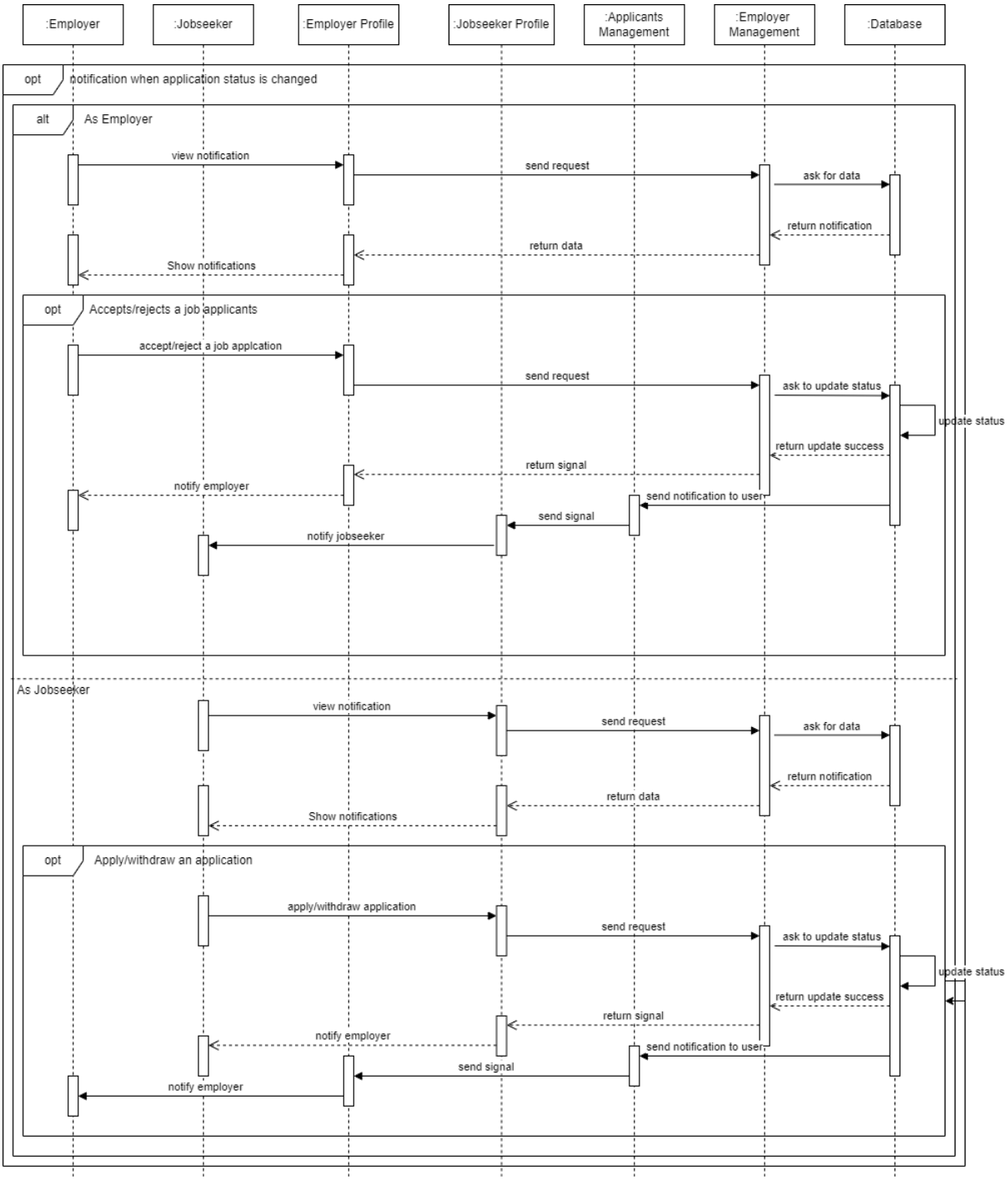
8.1 Updated Use Case Diagram



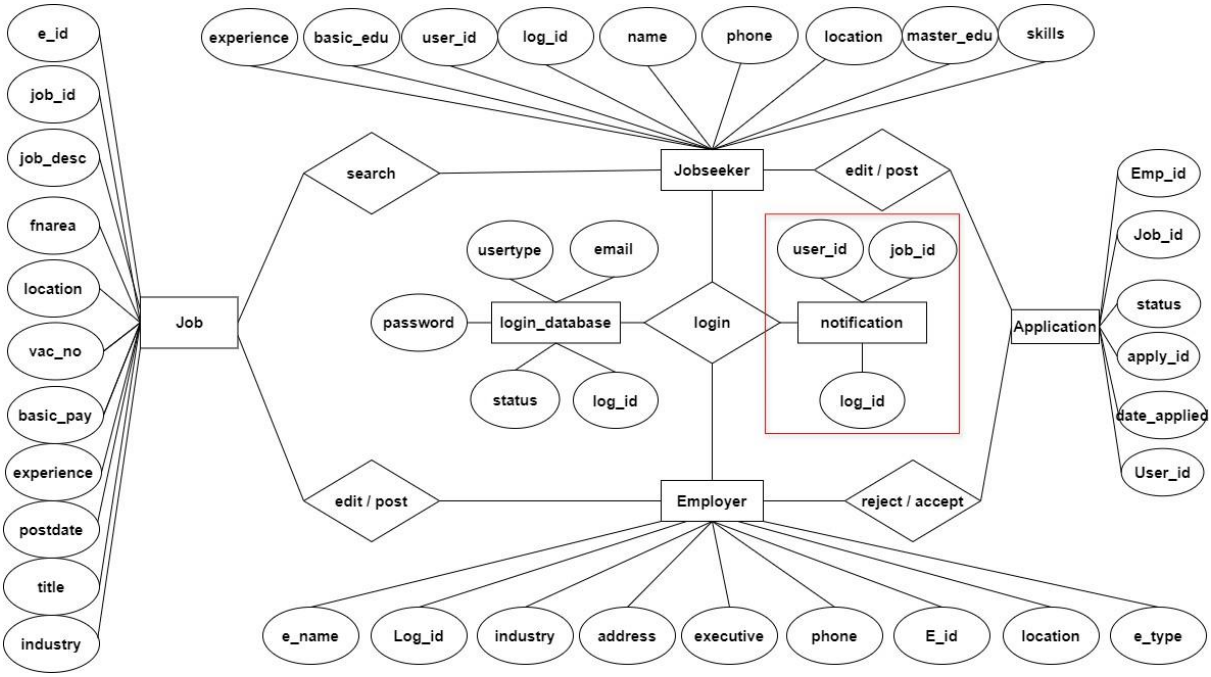
8.2 Updated Use Case Analysis

Name	Notify job application status
Description	This use case is for notifying both employers and jobseekers of the status of job applications.
Data Return	Pop up objects containing strings
Trigger	Users must click on "Notification" in the menu task bar.
Pre-conditions	Users must register as either employers or jobseekers to receive notifications related to their status
Post-conditions	Employer will receive notification when there are new applicants or when there are no vacancies available. Jobseeker will receive notification when there application status changed
Basic Flow	1. The Employer or jobseeker click on the notification in menu task bar. 2. The employer or jobseeker will able to see the latest change in application status or job posted.
Alternative Flow	1. When there are changes in application status or job posted the notification button on the task bar will have a red dot and the number representing the new notification 2. Employer and jobseeker can click on notification to see the latest change in application status or job posted.

8.3 Updated Sequence Diagram

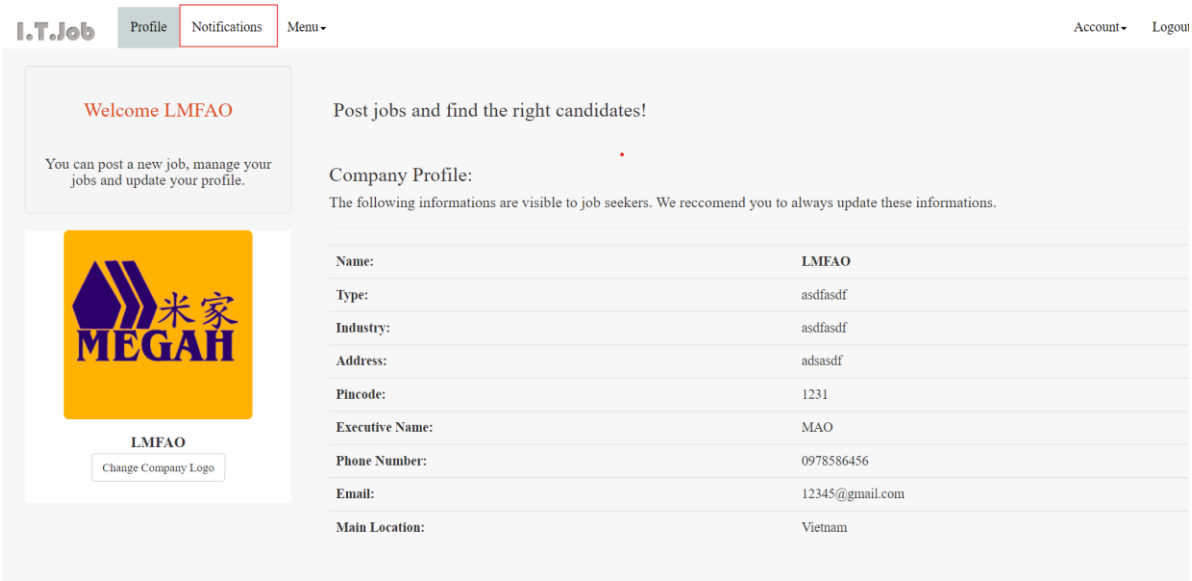


8.3.1 Updated ER Diagram

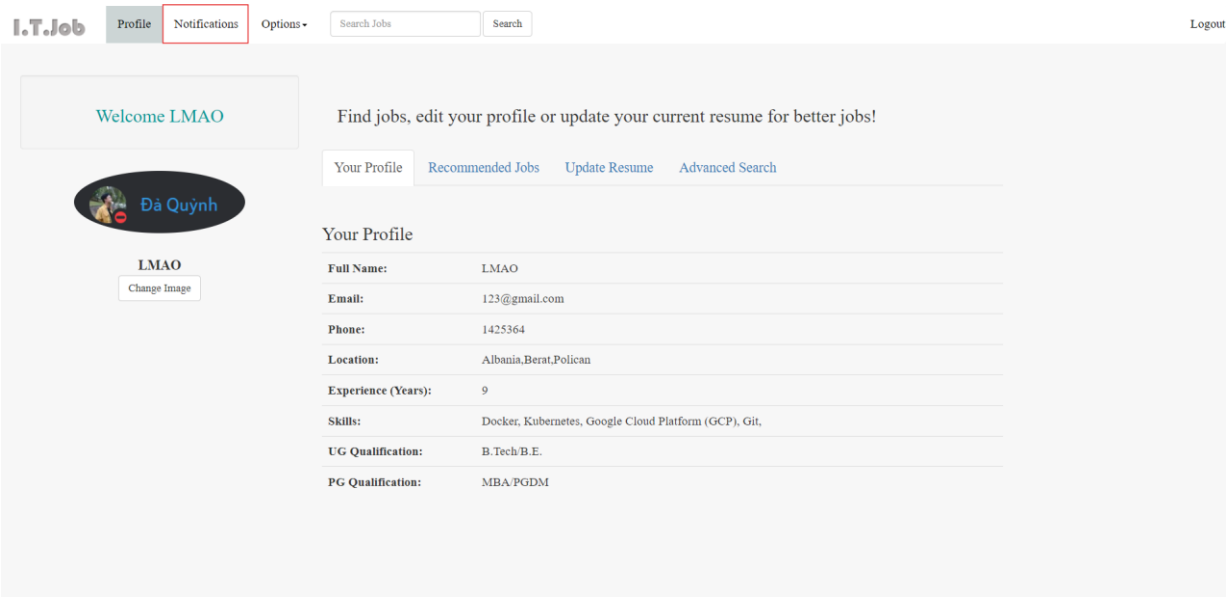


8.4 Updated UI

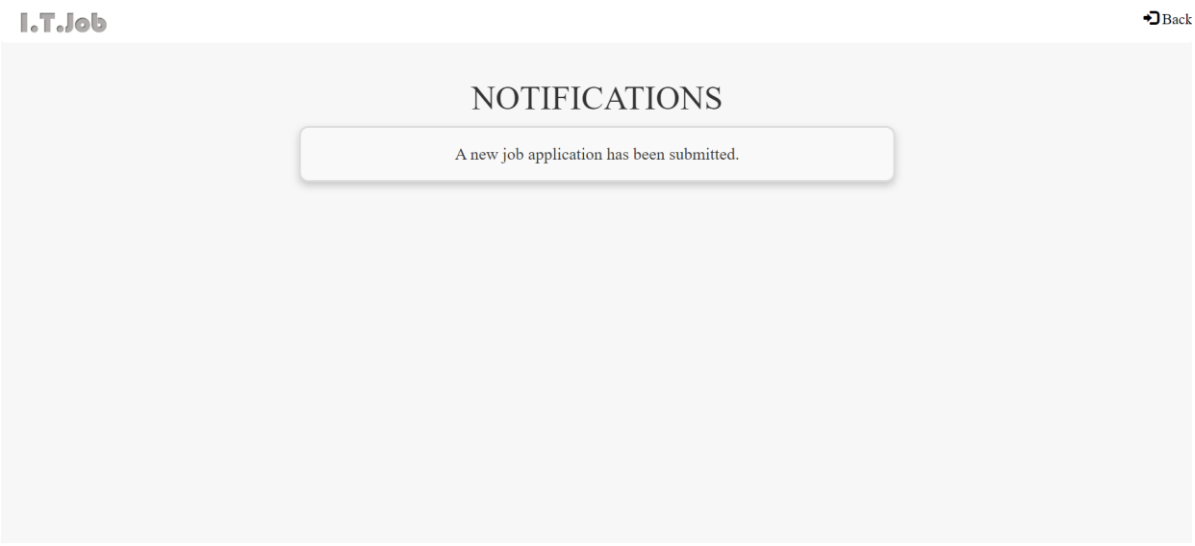
8.4.1 Updated Employer Page



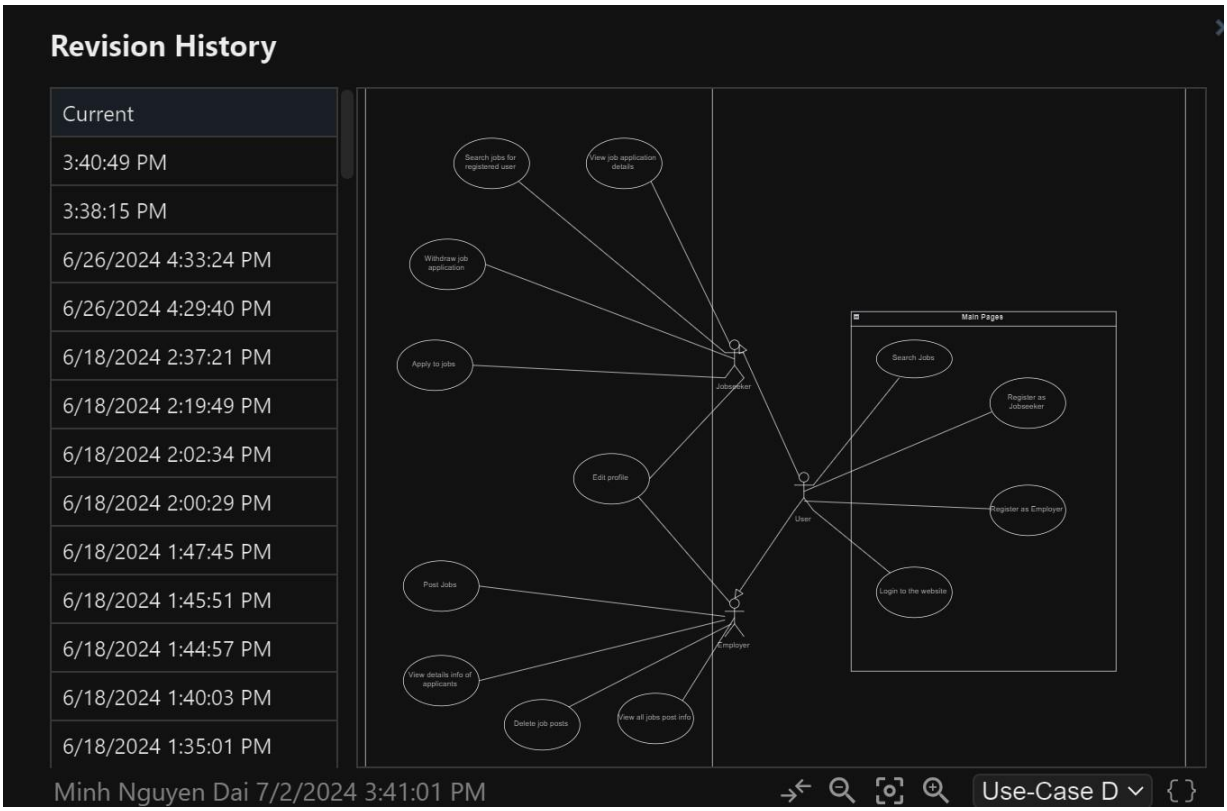
8.4.2 Updated Jobseeker Page



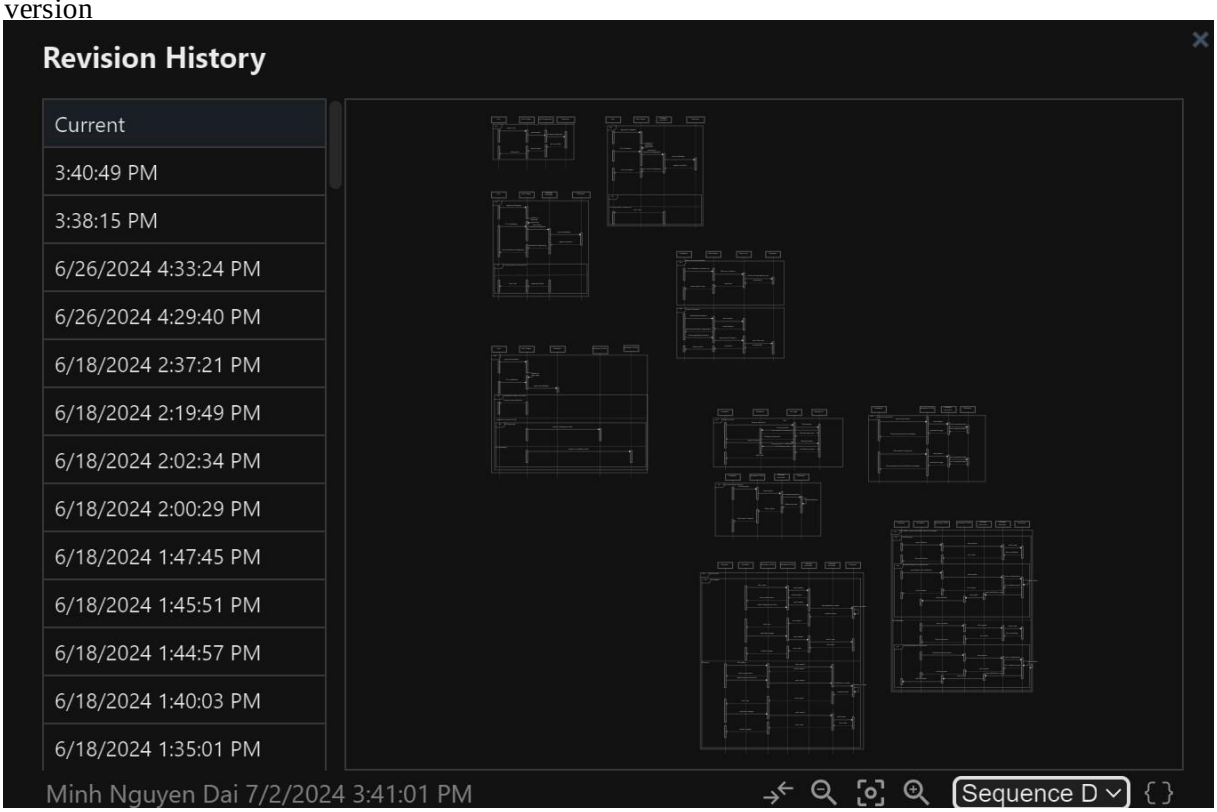
8.5 Updated Notification Page



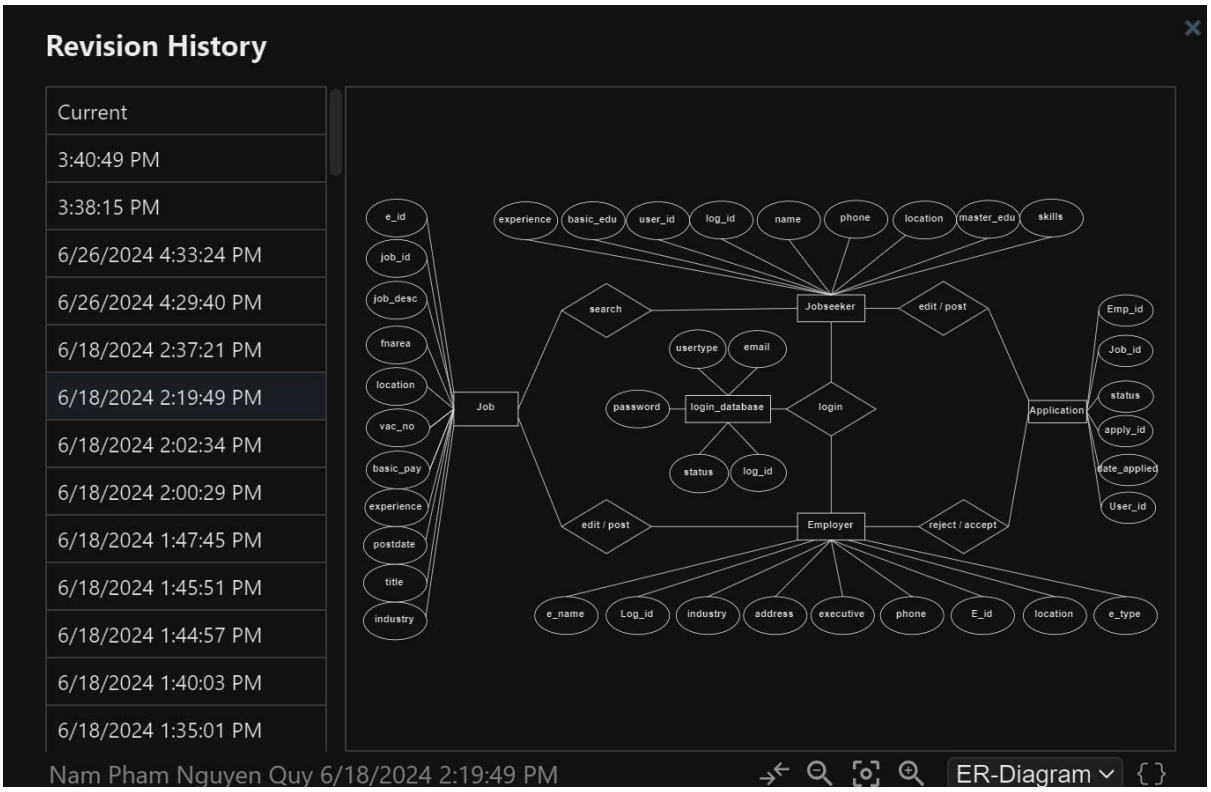
8.6 Revision History



Our previous version use case diagram dont have the new use case but we added it in our current version



Same goes for our sequence diagrams we have added 1 more diagram to our whole project



In the ER diagram you can see that our old version does not have the notfication entities