IN-HOUSE JOB PORTAL

AVINASH PATHROL

A project report submitted in conformity with the requirements
for the degree of Master's of Science in Information Technology

Department of Mathematical and Physical Sciences
Faculty of Graduate Studies
Concordia University of Edmonton

IN-HOUSE JOB PORTAL

AVINASH PATHROL

Approved:

Rossitza Marinova, Ph.D.

Supervisor Date

Committee Member Name, Ph.D.

Committee Member Date

Alison Yacyshyn, Ph.D.

Dean of Graduate Studies Date

In-house Job Portal Web Application

Avinash Pathrol

Master of Science in Information Technology
Department of Mathematical and Physical Sciences
Concordia University of Edmonton
2022

Abstract

The goal of this portal is to connect to industries and act as an online recruitment to support students to find the right IT job after/during graduation. This project addresses the gap between the Job Seeker and the Recruiter. This is done by taking into consideration the details provided by both, the Job Seeker and the Recruiter, and, by applying a variety of different filters in order to cater to each and everyone's individual needs and wishes. The methodology implemented is an effective model for the sole purpose of a web portal creation. The messaging system is quite an important feature which will be implemented in the coming future within the project in order to keep the involved parties informed about their status in the Job Portal, telling them everything such as – company to which they have applied, application status, designation, department. Since, the websites are, nowadays, accessed via a variety of different devices such as desktops, laptops, tablets, mobile devices, etc., using Bootstrap enables easy compatibility with all the above mentioned devices with ease.

Keywords: —Knowledge sharing, web portal, job portal, online recruitment.

Contents

List of Tables

List of Figures

Listings

1 Introduction

Unemployment is one of the serious social issues faced by both developing and developed countries. For example, in Europe the rate of unemployment has been increasing rapidly since the 1970's. Dorn and Naz [1] mentioned that one of the reasons for this problem is the unfair distribution or lack of information on job opportunities so people are unable to know the new job vacancies. It means that there are some jobs available, but jobseekers do not have access to that information. An efficient search of the internet might help to jobseekers in their job hunt. There are some web portals that provide an efficient way to search the web for online information on job vacancies for jobseekers [2]. Today, the internet has changed many aspects of our life, such as the way we look for jobs [3].

If one person wants to find a new job, they can open a web browser to send the resume and receive an e-mail. Online recruitment has become the standard method for employers and jobseekers to meet their respective objectives. The employers upload the job offerings in to the job portals. Online recruitment has been accepted not only by most large companies but also the small ones. The organizations send information or jobs vacancies for posting on the portals and communicate with the applicants via the Internet and Email. Gangle [4] defined the concept of online recruitment or e-recruitment as the use of Internet to search for jobs which have been advertised electronically. Thus, the employers advertise the job opportunities, save the resume and curriculum vitae (CV) of applicants, contact the jobseekers who are qualified, online.

Today, the Internet has become one of the key methods for getting information relating to job vacancies. Large institutions, like universities include information on career prospects in their websites which are also linked to recruitment sites. The rest of this research is arranged as follows: First of all, the paper presents a literature review including the job procurement: old and new ways, importance of job portals and features of job portals, and then it argues the methodology continues by data analysis and discussion, the last part brings the conclusion.

Portals have different applications or services to solve various problems. One of the main purposes of web portals is to allow information sharing over the Internet. This need can be addressed through a knowledge portal which must contain sufficient data and information about the requirements of the Job Seekers. Today, The Internet has changed many aspects of our life, such as the way we look for jobs.

Considering the aforementioned arguments, the information flow in the online labour market is far from optimal. A large number of online job portals have sprung up, dividing the online labour market into information islands and making it close to impossible for a job seeker to get an overview of all relevant open positions. Their strong market position, as the prime starting point for job seekers, allows job portals to charge employers high fees for publishing open positions. Due to these costs employers publish their job postings only on a small number of portals, which prevents

the offers from reaching all qualified applicants. Employers often receive a large number of applications for an open position, due to the strained situation of the labour market. The costs of manually preselecting potential candidates have risen and employers are searching for means to automate the preselection of candidates.

The rest of this research is arranged as follows: First of all, the paper presents a literature review including the job procurement: old and new ways, importance of job portals and features of job portals, and then it argues the methodology continues by data analysis and discussion, the last part brings the conclusion.

The aim of this project is to add to a stage for the graduates to have the job ideas of their related field. Through this application recruiters or additionally universities can instruct their students, to get the jobs. This web application gives an approach to take and apply for the available jobs quick and easily.Current students and recent graduates will discover this application so fascinating that they can easily search for all the available jobs across the country. This Application is as necessary as well as important in every aspect. One of the greater difficulties that the general universities faces is guaranteeing that students get job(s) properly, so that they may build up their aptitudes. The target of this is to outline a framework that will help students and applicants in getting new and better job(s).

2 Literature Review

2.1 Job Procurement: Old and New Ways

Job seeking usually involves different ways to look for jobs such as through personal contacts, direct telephone calls to employers, job agency office, scanning online job listings, etc. [3]. Before the Internet, became widely used as a method of seeking jobs, jobseekers spent a lots of time using various methods to look for job openings. Today, jobseekers use online methods which are very convenient and save a lot of time. Galanaki [5] lists the following methods to be the traditional (old) ways for recruitment:

- Employment recruitment agencies;
- Job fairs;
- Advertising in the mass media such as newspapers;
- Advertisement in television and radio;
- Management Consultants;
- Existing employee contacts;
- chools colleges or universities students services department;
- Workers or professional referrals.

These old job seeking methods are too slow, stressful, challenging and also lack quality [6]. In addition, the applicants have to consider the cost and the amount of time to get the information they need, and other preparations they have to make. Finding all available job vacancies is a main step at in the job-seeking process. The Internet is now a powerful tool that jobseekers can use. Today, there are many sites that advertise job positions to be filled by people with certain skills in various fields. The Internet plays an important role in the area of human resource planning and development. Most planning and development organizations are now using computer technology and the Internet for staff recruitment. It should be noted that although the Internet has facilitated the process of job-seeking, it has not replaced the traditional methods, completely.

2.2 Importance of Job Portals

In the age of technology, the Internet has become the main source of information for jobseekers. Large corporations, institutions, and universities include information on career prospects on their websites. These websites or portals provide a search engine to access information on job opportunities [7]. Sulaiman and Burke [8] found that most employers are keen to use online recruitment methods of getting staff.

He mentioned that online recruitment methods have the ability to identify the best applicants. That is the reason why more developed countries such as Malaysia have started to use online job portal as an important way to recruit people to fill job vacancies. A study done in 2006, found that 21% of internet users in the EU used the web to search for jobs or to send job applications. In 2007, this had increased to 67% for unemployed people [5].

Most companies publish their job vacancies on their website, or use online jobsites. These methods result in great cost savings. Mochol and Nixon [9] stated that the use of semantic web technology gives market transparency, higher speed of procurement but reduced transaction cost. Today, the Internet is used for a large number of business transactions. People find the Internet to be an effective communication tool. In a report in 2005, it was found that 90% of jobseekers in Germany use the internet to look for jobs. A reason for this high rate is that applicants are young and highly qualified and use the internet a lot, and many companies publish their job opportunities online and via their portal. Job portals are the starting point of jobseekers when searching for jobs.

Thus, some job portals charge employers high fees to publish information on job vacancies. In spite of this, many employers still continue to advertise or publish information on job opportunities on job portal, but limit this in order to keep costs down. Many employers still believe that a jobseeker will visit job portals when searching for job vacancies [10]. A good job portal can also support knowledge sharing among the members. The number of online job portals continues to increase. It is believed that three quarters of who are searching for jobs, use the internet and

online portals. Paradamean [4] stated that online recruitment has the following advantages: employers can identify a large number of eligible job seekers and get their information easily.

It means that organizations can extend the search domain, hence, they have better prospect of selecting the most qualified candidates. Internet provides employers a way to attract a higher number of candidates, especially, those who fulfill the job requirements. With online recruitment, people have access to the job information from anywhere in the world, while with the newspaper, information is disseminated at local level. One key aspect of job portal is the cost. Companies spend less to publish or advertise job vacancies on the portals or websites, as compared to the use of other media such as newspaper or job fairs. Furthermore, online recruitment is very fast, and saves time. Once the employers upload the job vacancy on the portal, the jobseekers are able to view it and send in their resume. Therefore, cost and time savings are two significant advantages of job portals.

2.3 Features of Job Portals

One of the ways to improve employment mobility is to provide online job offer services. Online job portals can help jobseekers as they contain all required information about available vacancies in a single point. Such portals enhance efficiency in job recruitment as applicants can match their qualifications and skills to the requirements of employers. Generally, searching for jobs on the internet involves a process of information collecting because the jobseeker gathers information contained in the job portals, during the search.

A good job portal shares information and experiences with its members/users. This saves time and efforts. Job openings requirements can be matched to an applicant's qualification and skills. In this way, job portals return not only the precise matches but also return the most similar match.

Online job portals should have quite similar characteristics that include: an online searchable database of positions for job searcher; facilities to send CVs to the website; email alerts of jobs which match the users profile; extra instruction, for example, about working in foreign countries or career guidance; the capability to manage job applications; employers must have the ability to publish and manage job positions, search the CV database; and have online contact with potential jobseekers.

3 Technical Frameworks

Web application development is not the same as before, even if it is a few years back. Today, there are so many options, and strangers are often confused about what is best for them. There are many options not only for a wide stack (various tiers or technology used), but also tools that help improve overall experience of web or software development.

We have used MongoDB, Express, ReactJs and NodeJs which is known as MERN stack because it is excellent for building a complete web system. The four components of the MERN stack (MongoDB, Expresses, ReactJs and NodeJs) and how well they work together, their beauty as a complete stack in web design. We have focused exclusively on the functions of these four MERN stack technologies and how they are applied to web development for project such as online job portals.

MERN is a acronym used to describe a specific set of JavaScript-based technologies used in the web application development process , It is designed with the idea of making the development process as smooth as possible. All of these factors play a very important role in the process of web application development. All of this provides the final framework for the engineers in which they work. Today around the world are working to improve the user interface of the app and to improve the developer process of building applications to implement projects and development requirements within a set deadline. Since MERN is based on JavaScript, developers only need to know one coding language, making things a million times easier.

3.1 MERN - FRONTEND

3.1.1 ReactJS

ReactJs is a JavaScript enabled library designed to build user web resources. React Virtual DOM is completely memory and is a web DOM browser. As a result, when we designed the React section, we do not write directly to the DOM; instead, we write the tool that the response will turn into a DOM. React is built around objects, not templates like other frames. The section can be created with the function of the Class React object, the first place when you access the library.

3.1.2 ReactJS Performance

When looking at the performance of a React or other framework it is important to remember that in the end, it is all JavaScript scripts the main purpose of these frameworks is to provide a better framework for UI development than to achieve high levels of efficiency in any given environment. Any pre-trip request can be reduced to its base level for high-performance fraud, but the result may be poorly formatted. Because all JavaScript location points eventually use the same base of the JavaScript API, the difference in performance boils how much additional code the framework surrounding those tools .

3.1.3 Reasoning behind the use of ReactJS

Because part of the front-end app is straightforward and does not have a functionality that is different enough for a single frame to be allowed to run, the selection is greatly reduced to personal preference. ReactJs has some advantages compared to other previous frameworks, such as a faster learning curve, support and future

development programs from the organization (Facebook), and strong documentation that has made it an easy-to-read and useful service framework.

3.2 MERN - BACKEND

3.2.1 NodeJS

NodeJS is a software framework that helps create unique network applications and events. It includes built-in HTTP libraries that allow developers to create their own web server and additionally create awesome web applications. NodeJs creates event and event managers for all applications. When I/O performance occurs, the corresponding handle is lined up to perform and the retrieval function exits the event after the completion of the I/O function. Certain I/O functions are currently running outside the server event line. Therefore, NodeJs performs I/O functions humbly and does not interfere with any documentation, allowing the event loop to respond to other requests. Node module provides a public API (Application Programming Interface) that can be used after the module is installed in an existing script. Basic modules come with Node installation and are synchronized when the Node cycle begins [4].

3.2.2 ExpressJS

ExpressJs is an open-source server format written entirely in JavaScript, designed specifically for use with NodeJs. Due to its reliability and widespread use Express has become a standard Node software. Express in relation to NodeJs can be considered as similar to what Ruby on Rails is on Ruby. Express is designed to work and is naturally small, placing only a thin layer over Node web features. Due to its light weight design and standard adjustment, Express serves as the basis for some JavaScript components including feathers, KeystoneJS, Kraken and Sails, which are designed for certain types of applications unlike Express, which only provides powerful performance standard server features such as route navigation, HTTP cache and template view Power efficiency is one of the most important requirements of this application, we show battery status performance. Express is a "server-side" or "back-end" framework. It is not comparable to customer-side organizations such as React, Angular and Vue. It can be used in conjunction with those frameworks to build a complete application.

3.2.3 MongoDB

There are many items under the MongoDB umbrella. MongoDB stores data in flexible scripts, such as JSON. The document model is mapped to objects in your app code, making data easier to work with. MongoDB is a database that is distributed in its context, high availability, horizontal scale and distributed locations are built-in and easy to use. MongoDB directed to text, no database following. MongoDB is used for its flexibility, Flexible Query Model, Native Aggregation, Schema.

Characteristics of MongoDB :

- General Purpose database;
- Flexible schema design;
- Scalability and Load balancing;
- Aggregation Framework;
- Native replication;
- Security features;
- JSON;
- MapReduce.

3.2.4 Working of MongoDB

Documents store data with the help of key-value pairs. Data stored in JSON format and the backend MongoDB converts the binary data known as BSON. Now, all of these connections are stored in data. A collection is a group of documents. These collections are stored in the MongoDB database. This makes fetching of data extremely efficient than other databases.

3.2.5 Application of MongoDB

- Internet of Things;
- Mobile Application;
- Real-time analysis;
- Personalization;
- Catalog management;
- Content management.

Companies such as Toyota, CISCO, Verizon, and Google easily use MongoDB as their data management system.

4 Application Work-flow

This is the Landing page of the website which serves as the main page for recruiter as well as applicants as soon as they sign-up and login into their accounts.

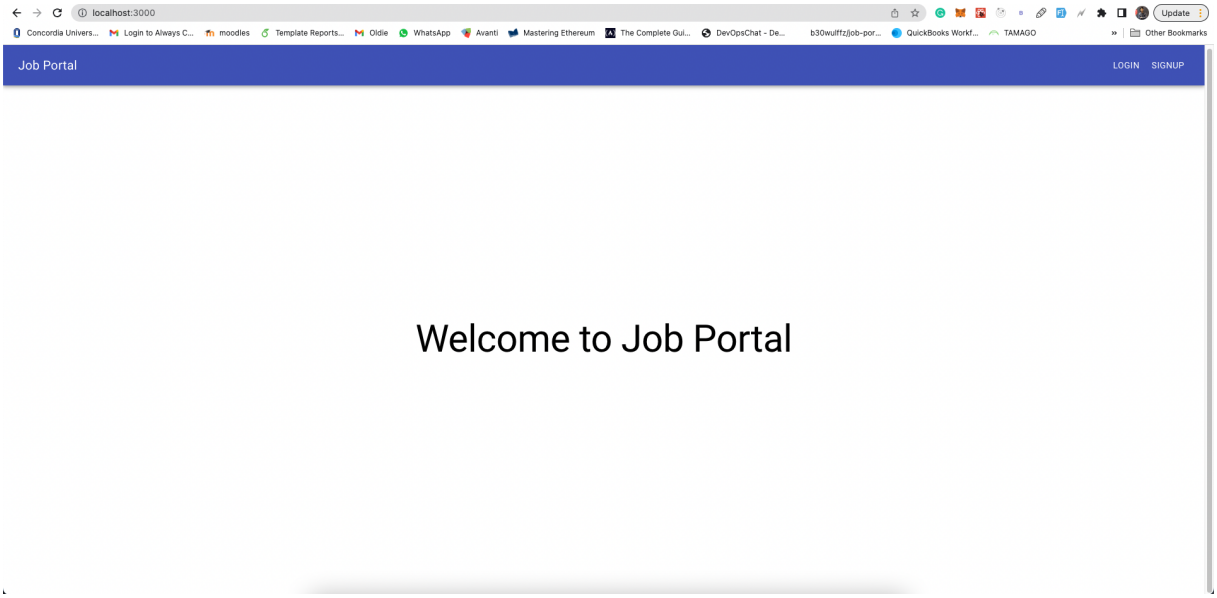

Figure 1: Application Landing Page

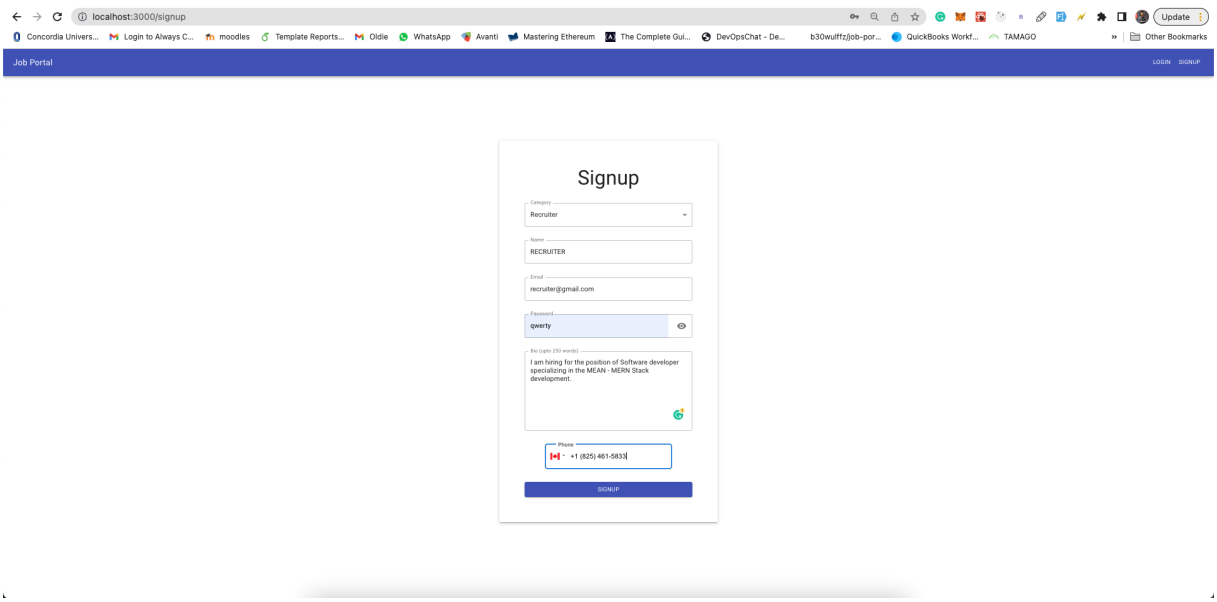

Figure 2: Recruiter Sign-up Module

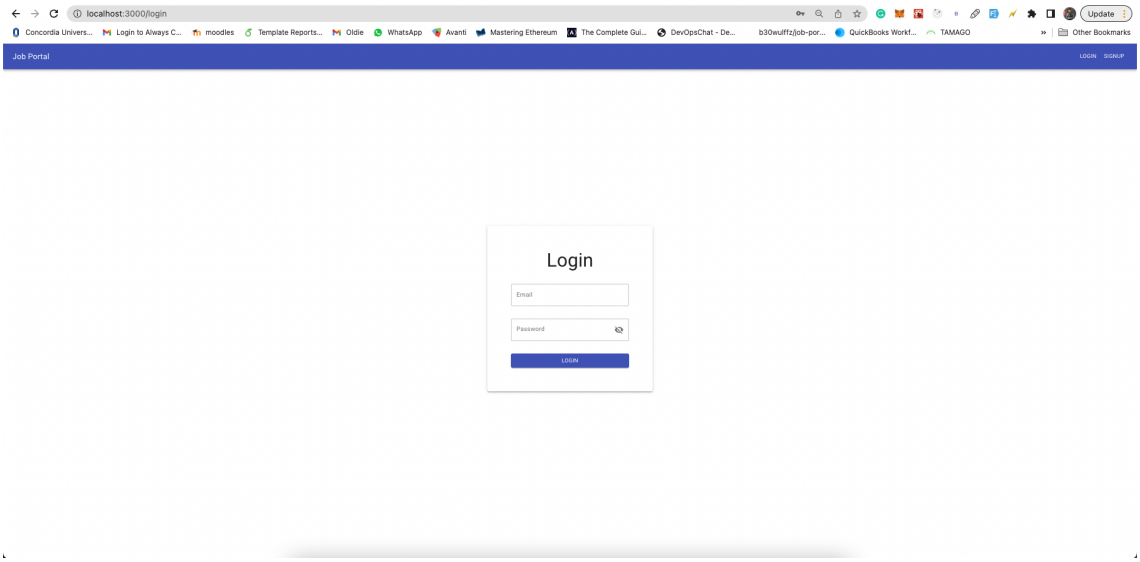

Figure 3: Recruiter Login Module

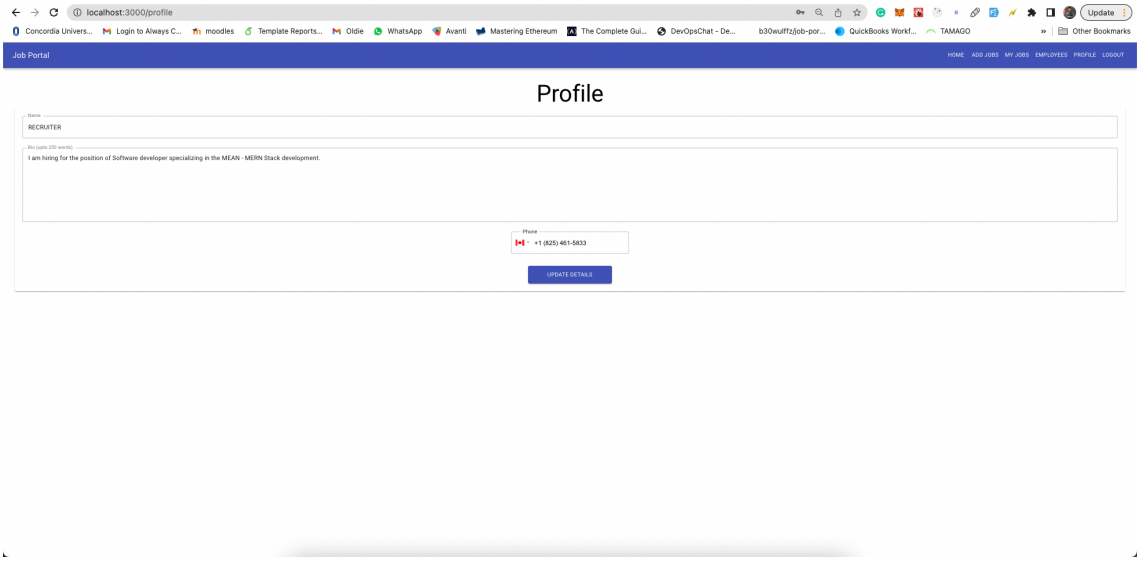

Figure 4: Recruiter Profile Module

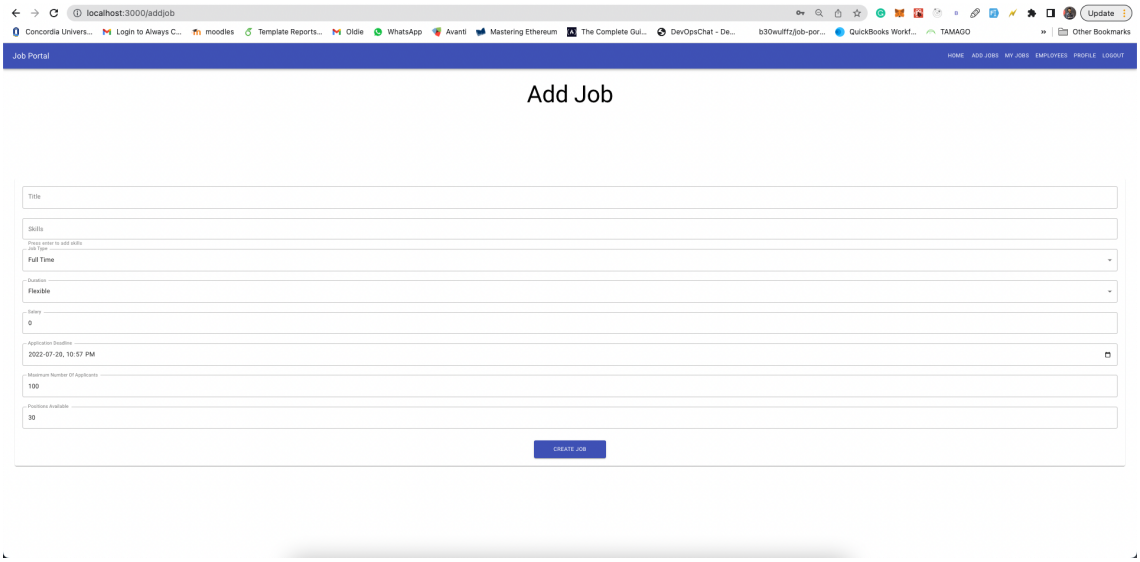

Figure 5: Recruiter Module to Add Jobs

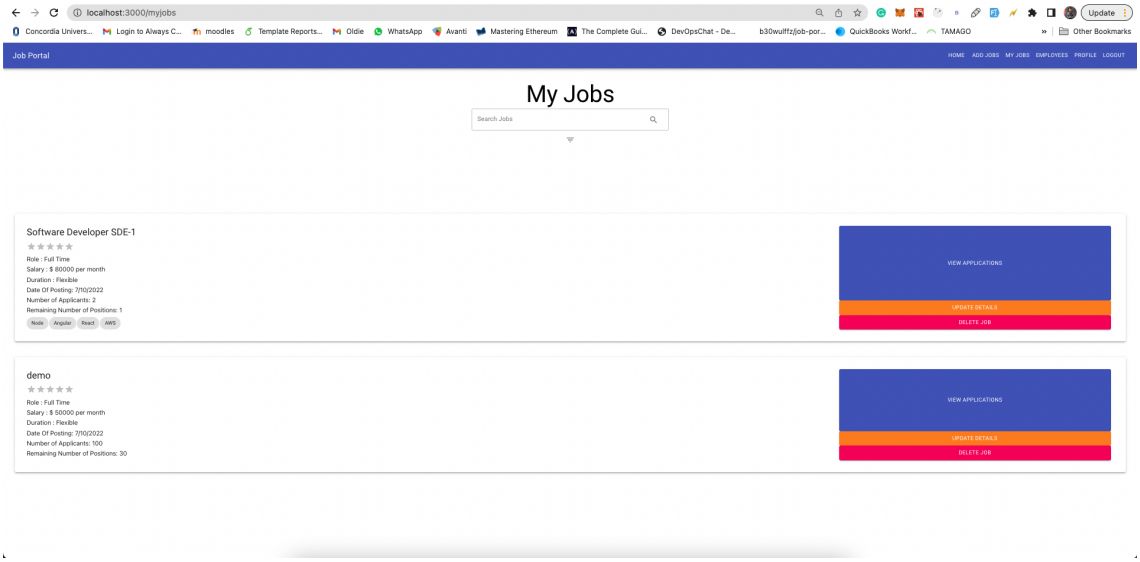

Figure 6: Recruiter Module which displays all the jobs posted by the recruiter

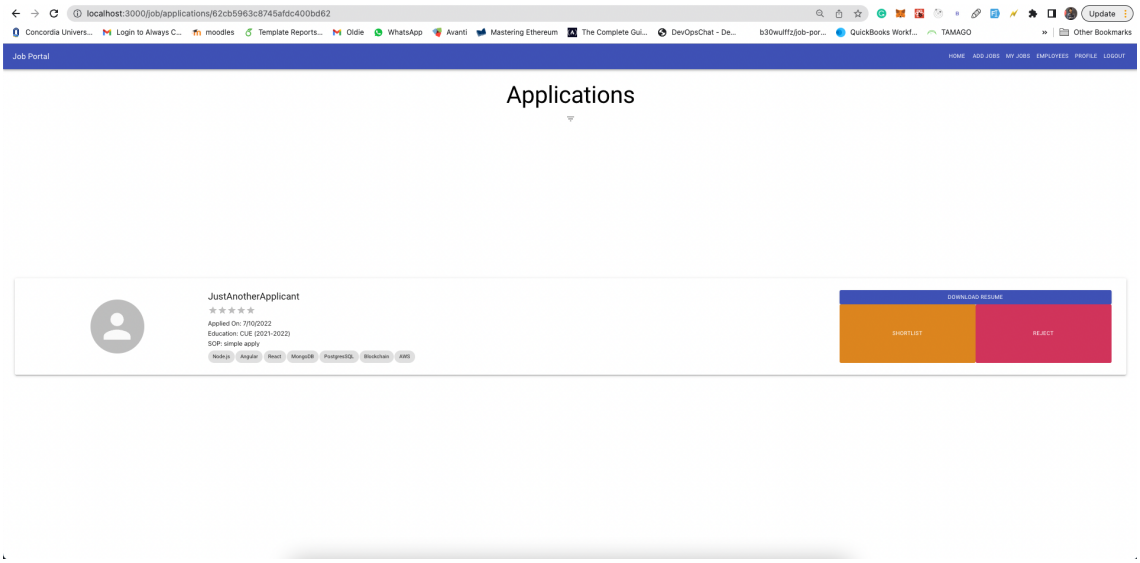

Figure 7: Recruiter module which enables Decision Making

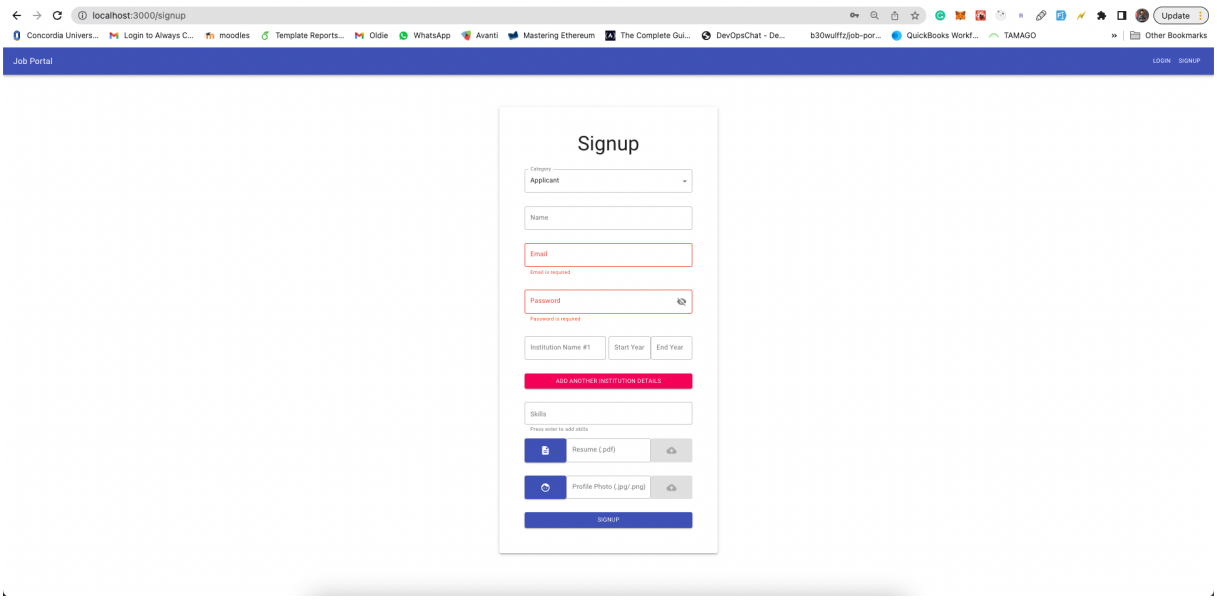

Figure 8: Applicant Sign-up Module

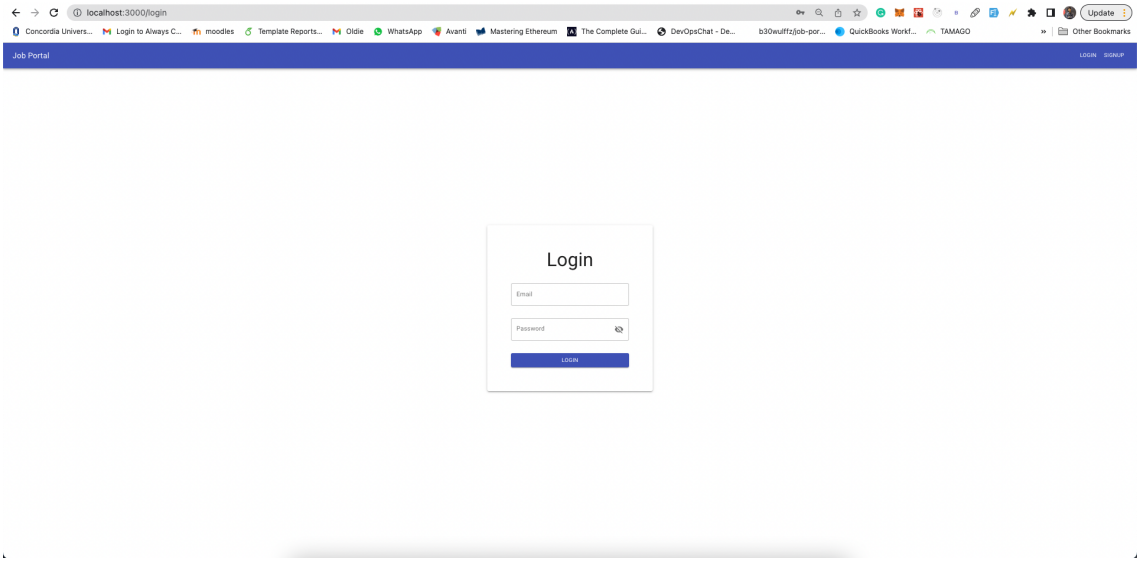

Figure 9: Applicant Login Module

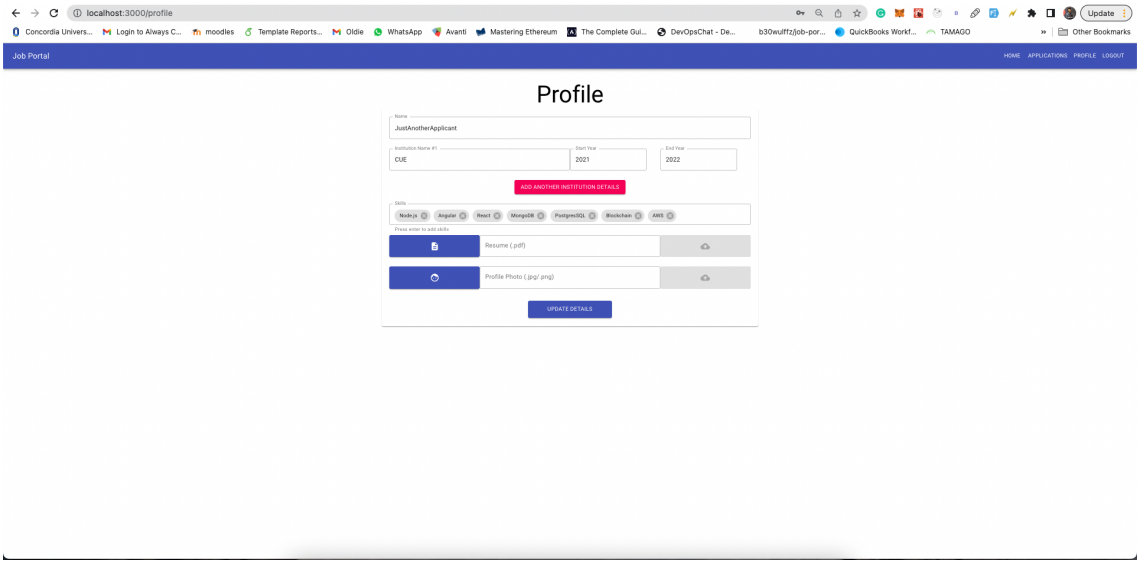

Figure 10: Applicant's Profile Section

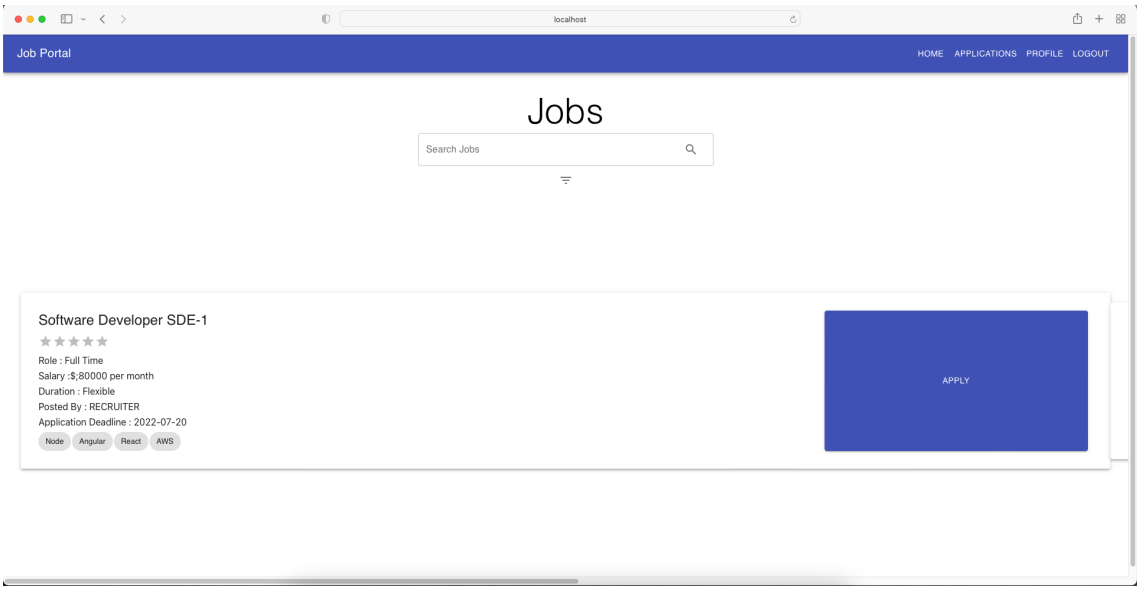

Figure 11: Job Listing on the Applicant Home page

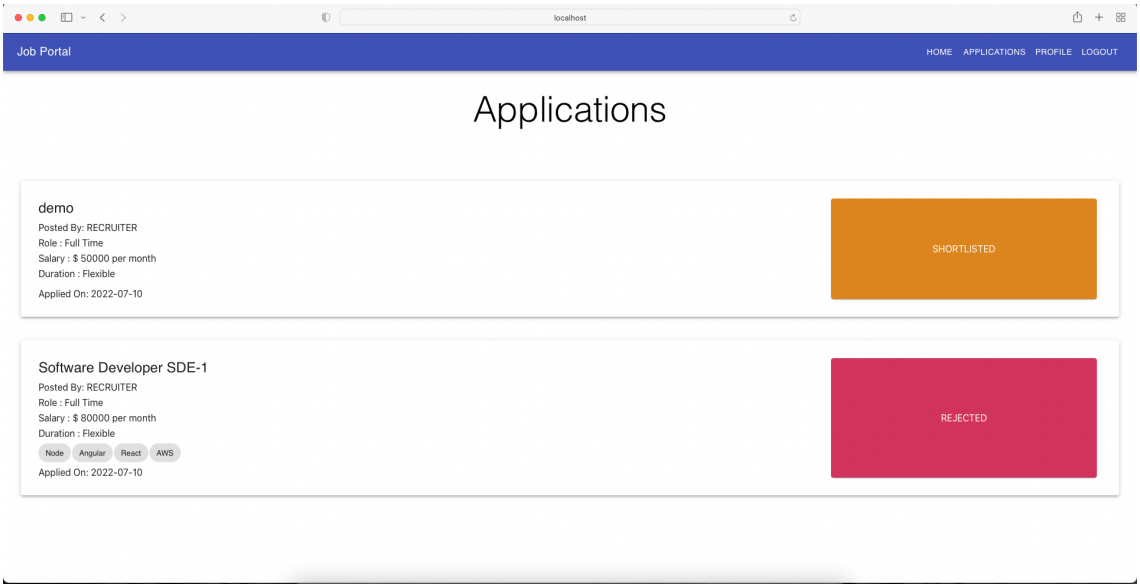

Figure 12: Applicant Job Application Decisions.

5 Appendix / Code

5.1 Login

This Code shows the Login module which is used on the website which enables the users and the recruiters to sign into their accounts.

React Login Module code is shown in the Listing 1.

Listing 1: React Login Module code

```
<!-- Login.js -->
<div class="login-wrapper" fxLayout="row" fxLayoutAlign="center center">
<mat-card class="box">
<mat-card-header>
<mat-card-title>Log in</mat-card-title>
</mat-card-header>
<form class="example-form" [formGroup]="LoginForm">
<mat-card-content>
<br />
<br />
<br />
<mat-form-field class="form-group example-full-width">
<input
   matInput
   class="form-control"
   name="username "
   type="text"
   placeholder="Email"

   formControlName="username"

   formControlName="email"
   [errorStateMatcher]="matcher"

   oninput="this.value = this.value.toLowerCase()"

/>
<mat-error *ngIf="
    LoginForm.get('email')?.hasError('email') ||
    LoginForm.get('email')?.hasError('required')
  ">
   Please enter valid email address
</mat-error>
</mat-form-field>
<br />
```

```
<br />
<br />
<mat-form-field class="form-group example-full-width">
<input
   class="form-control"
   name="password"
   type="text"
   matInput
   placeholder="Password"
   formControlName="password"
   [errorStateMatcher]="matcher"
   [type]="hidePassword ? 'password' : 'text'"
/>
<mat-icon matSuffix (click)="hidePassword =
!hidePassword">{{hidePassword ?
   'visibility_off' : 'visibility'}}</mat-icon>

<mat-error *ngIf="LoginForm.get('password')?.hasError('required')">
   Please enter a password
</mat-error>
<div class="signup">

   <h4><a href [routerLink]="'/forgotpasswordemail'">
   Forgot Password</a></h4>
</div>
</mat-form-field>
<br />
<br />
<br />
</mat-card-content>
<button
mat-stroked-button
(click)="onSubmit()"
type="submit"
value="Login"
class="btn-block">
Log in
</button>
<span class="errorOnSubmit" *ngIf="showError">
{{errorMessage}}
</span>
<br />
<br />
<!-- <div class="forgotpassword">
```

```
<h4><a href [routerLink]="'/forgotpasswordemail'">
Forgot password? </a></h4>
</div> -->
<div class="signup">
<h3> Don't have an account?   </h3> <br>
<h4><a href [routerLink]="'/register'"> Create Account </a></h4>
</div>
<!-- <div class="signup">
<h3> Don't have an account?   </h3> <br>
<h4><a href [routerLink]="'/forgotpasswordemail'">
Forgot Password</a></h4>
</div> -->
</form>
</mat-card>
<!-- <div class="signup">
<h3> Don't have an account?   </h3> <br>
<h4><a href [routerLink]="'/register'"> Sign Up </a></h4>
</div> -->
</div>
```

5.2 Signup

This is the sign-up module of the front-end which is built using ReactJS which internally works around JSX. This module enables the recruiter as well as the applicant to sign-up for the portal to access the services. React Sign-up Module code is shown in the Listing 2.

Listing 2: React Sign-up Module code

```
\begin{lstlisting}
import { useState, useContext } from "react";
import {
Grid,
TextField,
Button,
Typography,
makeStyles,
Paper,
MenuItem,
Input,
} from "@material-ui/core";
import axios from "axios";
import { Redirect } from "react-router-dom";
import ChipInput from "material-ui-chip-input";
import DescriptionIcon from "@material-ui/icons/Description";
```

```
import FaceIcon from "@material-ui/icons/Face";
import PhoneInput from "react-phone-input-2";
import "react-phone-input-2/lib/material.css";

import PasswordInput from "../lib/PasswordInput";
import EmailInput from "../lib/EmailInput";
import FileUploadInput from "../lib/FileUploadInput";
import { SetPopupContext } from "../App";

import apiList from "../lib/apiList";
import isAuth from "../lib/isAuth";

const useStyles = makeStyles((theme) => ({
body: {
padding: "60px 60px",
},
inputBox: {
width: "400px",
},
submitButton: {
width: "400px",
},
}));

const MultifieldInput = (props) => {
const classes = useStyles();
const { education, setEducation } = props;

return (
<>
{education.map((obj, key) => (
<Grid
item
container
className={classes.inputBox}
key={key}
style={{ paddingLeft: 0, paddingRight: 0 }}
>
<Grid item xs={6}>
<TextField
label={`Institution Name #${key + 1}`}
value={education[key].institutionName}
onChange={(event) => {
const newEdu = [...education];
```

```
newEdu[key].institutionName = event.target.value;
setEducation(newEdu);
}}
variant="outlined"
/>
</Grid>
<Grid item xs={3}>
<TextField
label="Start Year"
value={obj.startYear}
variant="outlined"
type="number"
onChange={(event) => {
const newEdu = [...education];
newEdu[key].startYear = event.target.value;
setEducation(newEdu);
}}
/>
</Grid>
<Grid item xs={3}>
<TextField
label="End Year"
value={obj.endYear}
variant="outlined"
type="number"
onChange={(event) => {
const newEdu = [...education];
newEdu[key].endYear = event.target.value;
setEducation(newEdu);
}}
/>
</Grid>
</Grid>
))}
<Grid item>
<Button
variant="contained"
color="secondary"
onClick={() =>
setEducation([
...education,
{
institutionName: "",
startYear: "",
```

```
endYear: "",
},
])
}
className={classes.inputBox}
>
Add another institution details
</Button>
</Grid>
</>
);
};

const Login = (props) => {
const classes = useStyles();
const setPopup = useContext(SetPopupContext);

const [loggedin, setLoggedin] = useState(isAuth());

const [signupDetails, setSignupDetails] = useState({
type: "applicant",
email: "",
password: "",
name: "",
education: [],
skills: [],
resume: "",
profile: "",
bio: "",
contactNumber: "",
});

const [phone, setPhone] = useState("");

const [education, setEducation] = useState([
{
institutionName: "",
startYear: "",
endYear: "",
},
]);

const [inputErrorHandler, setInputErrorHandler] = useState({
email: {
```

```
untouched: true,
required: true,
error: false,
message: "",
},
password: {
untouched: true,
required: true,
error: false,
message: "",
},
name: {
untouched: true,
required: true,
error: false,
message: "",
},
});

const handleInput = (key, value) => {
setSignupDetails({
...signupDetails,
[key]: value,
});
};

const handleInputError = (key, status, message) => {
setInputErrorHandler({
...inputErrorHandler,
[key]: {
required: true,
untouched: false,
error: status,
message: message,
},
});
};

const handleLogin = () => {
const tmpErrorHandler = {};
Object.keys(inputErrorHandler).forEach((obj) => {
if (inputErrorHandler[obj].required &&
inputErrorHandler[obj].untouched) {
tmpErrorHandler[obj] = {
```

```
required: true,
untouched: false,
error: true,
message: `${obj[0].toUpperCase() + obj.substr(1)} is required`,
};
} else {
tmpErrorHandler[obj] = inputErrorHandler[obj];
}
});

console.log(education);

let updatedDetails = {
...signupDetails,
education: education
.filter((obj) => obj.institutionName.trim() !== "")
.map((obj) => {
if (obj["endYear"] === "") {
delete obj["endYear"];
}
return obj;
}),
};

setSignupDetails(updatedDetails);

const verified = !Object.keys(tmpErrorHandler).some((obj) => {
return tmpErrorHandler[obj].error;
});

if (verified) {
axios
.post(apiList.signup, updatedDetails)
.then((response) => {
localStorage.setItem("token", response.data.token);
localStorage.setItem("type", response.data.type);
setLoggedin(isAuth());
setPopup({
open: true,
severity: "success",
message: "Logged in successfully",
});
console.log(response);
})
```

```
.catch((err) => {
setPopup({
open: true,
severity: "error",
message: err.response.data.message,
});
console.log(err.response);
});
} else {
setInputErrorHandler(tmpErrorHandler);
setPopup({
open: true,
severity: "error",
message: "Incorrect Input",
});
}
};

const handleLoginRecruiter = () => {
const tmpErrorHandler = {};
Object.keys(inputErrorHandler).forEach((obj) => {
if (inputErrorHandler[obj].required &&
inputErrorHandler[obj].untouched) {
tmpErrorHandler[obj] = {
required: true,
untouched: false,
error: true,
message: `${obj[0].toUpperCase() + obj.substr(1)} is required`,
};
} else {
tmpErrorHandler[obj] = inputErrorHandler[obj];
}
});

let updatedDetails = {
...signupDetails,
};
if (phone !== "") {
updatedDetails = {
...signupDetails,
contactNumber: `+${phone}`,
};
} else {
updatedDetails = {
```

```
...signupDetails,
contactNumber: "",
};
}

setSignupDetails(updatedDetails);

const verified = !Object.keys(tmpErrorHandler).some((obj) => {
return tmpErrorHandler[obj].error;
});

console.log(updatedDetails);

if (verified) {
axios
.post(apiList.signup, updatedDetails)
.then((response) => {
localStorage.setItem("token", response.data.token);
localStorage.setItem("type", response.data.type);
setLoggedin(isAuth());
setPopup({
open: true,
severity: "success",
message: "Logged in successfully",
});
console.log(response);
})
.catch((err) => {
setPopup({
open: true,
severity: "error",
message: err.response.data.message,
});
console.log(err.response);
});
} else {
setInputErrorHandler(tmpErrorHandler);
setPopup({
open: true,
severity: "error",
message: "Incorrect Input",
});
}
};
```

```
return loggedin ? (
<Redirect to="/" />
) : (
<Paper elevation={3} className={classes.body}>
<Grid container direction="column" spacing={4} alignItems="center">
<Grid item>
<Typography variant="h3" component="h2">
Signup
</Typography>
</Grid>
<Grid item>
<TextField
select
label="Category"
variant="outlined"
className={classes.inputBox}
value={signupDetails.type}
onChange={(event) => {
handleInput("type", event.target.value);
}}
>
<MenuItem value="applicant">Applicant</MenuItem>
<MenuItem value="recruiter">Recruiter</MenuItem>
</TextField>
</Grid>
<Grid item>
<TextField
label="Name"
value={signupDetails.name}
onChange={(event) => handleInput("name", event.target.value)}
className={classes.inputBox}
error={inputErrorHandler.name.error}
helperText={inputErrorHandler.name.message}
onBlur={(event) => {
if (event.target.value === "") {
handleInputError("name", true, "Name is required");
} else {
handleInputError("name", false, "");
}
}}
variant="outlined"
/>
</Grid>
```

```
<Grid item>
<EmailInput
label="Email"
value={signupDetails.email}
onChange={(event) => handleInput("email", event.target.value)}
inputErrorHandler={inputErrorHandler}
handleInputError={handleInputError}
className={classes.inputBox}
required={true}
/>
</Grid>
<Grid item>
<PasswordInput
label="Password"
value={signupDetails.password}
onChange={(event) => handleInput("password", event.target.value)}
className={classes.inputBox}
error={inputErrorHandler.password.error}
helperText={inputErrorHandler.password.message}
onBlur={(event) => {
if (event.target.value === "") {
handleInputError("password", true, "Password is required");
} else {
handleInputError("password", false, "");
}
}}
/>
</Grid>
{signupDetails.type === "applicant" ? (
<>
<MultifieldInput
education={education}
setEducation={setEducation}
/>
<Grid item>
<ChipInput
className={classes.inputBox}
label="Skills"
variant="outlined"
helperText="Press enter to add skills"
onChange={(chips) =>
setSignupDetails({ ...signupDetails, skills: chips })
}
/>
```

```
</Grid>
<Grid item>
<FileUploadInput
className={classes.inputBox}
label="Resume (.pdf)"
icon={<DescriptionIcon />}
// value={files.resume}
// onChange={(event) =>
//   setFiles({
//     ...files,
//     resume: event.target.files[0],
//   })
// }
uploadTo={apiList.uploadResume}
handleInput={handleInput}
identifier={"resume"}
/>
</Grid>
<Grid item>
<FileUploadInput
className={classes.inputBox}
label="Profile Photo (.jpg/.png)"
icon={<FaceIcon />}
// value={files.profileImage}
// onChange={(event) =>
//   setFiles({
//     ...files,
//     profileImage: event.target.files[0],
//   })
// }
uploadTo={apiList.uploadProfileImage}
handleInput={handleInput}
identifier={"profile"}
/>
</Grid>
</>
) : (
<>
<Grid item style={{ width: "100%" }}>
<TextField
label="Bio (upto 250 words)"
multiline
rows={8}
style={{ width: "100%" }}
```

```
variant="outlined"
value={signupDetails.bio}
onChange={(event) => {
if (
event.target.value.split(" ").filter(function (n) {
return n != "";
}).length <= 250
) {
handleInput("bio", event.target.value);
}
}}
/>
</Grid>
<Grid item>
<PhoneInput
country={"in"}
value={phone}
onChange={(phone) => setPhone(phone)}
/>
</Grid>
</>
)}

<Grid item>
<Button
variant="contained"
color="primary"
onClick={() => {
signupDetails.type === "applicant"
? handleLogin()
: handleLoginRecruiter();
}}
className={classes.submitButton}
>
Signup
</Button>
</Grid>
</Grid>
</Paper>
);
};

export default Login;
```

5.3 Homepage

This is the home page of the portal which displays data based on the access and the user's personal choices. For applicant this shows the jobs that are available in the job market whereas when it comes to recruiters, it displays the jobs that they have pushed live and enables them to modify details based on certain criteria. React Home Module code is shown in the Listing 3.

Listing 3: React Home Module code

```
import { useState, useEffect, useContext } from "react";
import {
Button,
Chip,
Grid,
IconButton,
InputAdornment,
makeStyles,
Paper,
TextField,
Typography,
Modal,
Slider,
FormControlLabel,
FormGroup,
MenuItem,
Checkbox,
} from "@material-ui/core";
import Rating from "@material-ui/lab/Rating";
import Pagination from "@material-ui/lab/Pagination";
import axios from "axios";
import SearchIcon from "@material-ui/icons/Search";
import FilterListIcon from "@material-ui/icons/FilterList";
import ArrowUpwardIcon from "@material-ui/icons/ArrowUpward";
import ArrowDownwardIcon from "@material-ui/icons/ArrowDownward";

import { SetPopupContext } from "../App";

import apiList from "../lib/apiList";
import { userType } from "../lib/isAuth";

const useStyles = makeStyles((theme) => ({
body: {
height: "inherit",
},
button: {
```

```
width: "100%",
height: "100%",
},
jobTileOuter: {
padding: "30px",
margin: "20px 0",
boxSizing: "border-box",
width: "100%",
},
popupDialog: {
height: "100%",
display: "flex",
alignItems: "center",
justifyContent: "center",
},
}));

const JobTile = (props) => {
const classes = useStyles();
const { job } = props;
const setPopup = useContext(SetPopupContext);

const [open, setOpen] = useState(false);
const [sop, setSop] = useState("");

const handleClose = () => {
setOpen(false);
setSop("");
};

const handleApply = () => {
console.log(job._id);
console.log(sop);
axios
.post(
`${apiList.jobs}/${job._id}/applications`,
{
sop: sop,
},
{
headers: {
Authorization: `Bearer ${localStorage.getItem("token")}`,
},
}
```

```
)
.then((response) => {
setPopup({
open: true,
severity: "success",
message: response.data.message,
});
handleClose();
})
.catch((err) => {
console.log(err.response);
setPopup({
open: true,
severity: "error",
message: err.response.data.message,
});
handleClose();
});
};

const deadline = new Date(job.deadline).toLocaleDateString();

return (
<Paper className={classes.jobTileOuter} elevation={3}>
<Grid container>
<Grid container item xs={9} spacing={1} direction="column">
<Grid item>
<Typography variant="h5">{job.title}</Typography>
</Grid>
<Grid item>
<Rating value={job.rating !== -1 ? job.rating : null} readOnly />
</Grid>
<Grid item>Role : {job.jobType}</Grid>
<Grid item>Salary :&#x24;{job.salary} per month</Grid>
<Grid item>
Duration :{"␣"}
{job.duration !== 0 ? `${job.duration} month` : `Flexible`}
</Grid>
<Grid item>Posted By : {job.recruiter.name}</Grid>
<Grid item>Application Deadline : {deadline}</Grid>

<Grid item>
{job.skillsets.map((skill) => (
<Chip label={skill} style={{ marginRight: "2px" }} />
```

```
))}
</Grid>
</Grid>
<Grid item xs={3}>
<Button
variant="contained"
color="primary"
className={classes.button}
onClick={() => {
setOpen(true);
}}
disabled={userType() === "recruiter"}
>
Apply
</Button>
</Grid>
</Grid>
<Modal open={open} onClose={handleClose} className={classes.popupDialog}>
<Paper
style={{
padding: "20px",
outline: "none",
display: "flex",
flexDirection: "column",
justifyContent: "center",
minWidth: "50%",
alignItems: "center",
}}
>
<TextField
label="Write SOP (upto 250 words)"
multiline
rows={8}
style={{ width: "100%", marginBottom: "30px" }}
variant="outlined"
value={sop}
onChange={(event) => {
if (
event.target.value.split(" ").filter(function (n) {
return n != "";
}).length <= 250
) {
setSop(event.target.value);
}
```

```
}}
/>
<Button
variant="contained"
color="primary"
style={{ padding: "10px 50px" }}
onClick={() => handleApply()}
>
Submit
</Button>
</Paper>
</Modal>
</Paper>
);
};

const FilterPopup = (props) => {
const classes = useStyles();
const { open, handleClose, searchOptions,
setSearchOptions, getData } = props;
return (
<Modal open={open} onClose={handleClose}
className={classes.popupDialog}>
<Paper
style={{
padding: "50px",
outline: "none",
minWidth: "50%",
}}
>
<Grid container direction="column" alignItems="center" spacing={3}>
<Grid container item alignItems="center">
<Grid item xs={3}>
Job Type
</Grid>
<Grid
container
item
xs={9}
justify="space-around"
// alignItems="center"
>
<Grid item>
<FormControlLabel
```

```
control={
<Checkbox
name="fullTime"
checked={searchOptions.jobType.fullTime}
onChange={(event) => {
setSearchOptions({
...searchOptions,
jobType: {
...searchOptions.jobType,
[event.target.name]: event.target.checked,
},
});
}}
/>
}
label="Full Time"
/>
</Grid>
<Grid item>
<FormControlLabel
control={
<Checkbox
name="partTime"
checked={searchOptions.jobType.partTime}
onChange={(event) => {
setSearchOptions({
...searchOptions,
jobType: {
...searchOptions.jobType,
[event.target.name]: event.target.checked,
},
});
}}
/>
}
label="Part Time"
/>
</Grid>
<Grid item>
<FormControlLabel
control={
<Checkbox
name="wfh"
checked={searchOptions.jobType.wfh}
```

```
onChange={(event) => {
setSearchOptions({
...searchOptions,
jobType: {
...searchOptions.jobType,
[event.target.name]: event.target.checked,
},
});
}}
/>
}
label="Work From Home"
/>
</Grid>
</Grid>
</Grid>
<Grid container item alignItems="center">
<Grid item xs={3}>
Salary
</Grid>
<Grid item xs={9}>
<Slider
valueLabelDisplay="auto"
valueLabelFormat={(value) => {
return value * (100000 / 100);
}}
marks={[
{ value: 0, label: "0" },
{ value: 100, label: "100000" },
]}
value={searchOptions.salary}
onChange={(event, value) =>
setSearchOptions({
...searchOptions,
salary: value,
})
}
/>
</Grid>
</Grid>
<Grid container item alignItems="center">
<Grid item xs={3}>
Duration
</Grid>
```

```
<Grid item xs={9}>
<TextField
select
label="Duration"
variant="outlined"
fullWidth
value={searchOptions.duration}
onChange={(event) =>
setSearchOptions({
...searchOptions,
duration: event.target.value,
})
}
>
<MenuItem value="0">All</MenuItem>
<MenuItem value="1">1</MenuItem>
<MenuItem value="2">2</MenuItem>
<MenuItem value="3">3</MenuItem>
<MenuItem value="4">4</MenuItem>
<MenuItem value="5">5</MenuItem>
<MenuItem value="6">6</MenuItem>
<MenuItem value="7">7</MenuItem>
</TextField>
</Grid>
</Grid>
<Grid container item alignItems="center">
<Grid item xs={3}>
Sort
</Grid>
<Grid item container direction="row" xs={9}>
<Grid
item
container
xs={4}
justify="space-around"
alignItems="center"
style={{ border: "1px solid #D1D1D1", borderRadius: "5px" }}
>
<Grid item>
<Checkbox
name="salary"
checked={searchOptions.sort.salary.status}
onChange={(event) =>
setSearchOptions({
```

```
...searchOptions,
sort: {
...searchOptions.sort,
salary: {
...searchOptions.sort.salary,
status: event.target.checked,
},
},
})
}
id="salary"
/>
</Grid>
<Grid item>
<label for="salary">
<Typography>Salary</Typography>
</label>
</Grid>
<Grid item>
<IconButton
disabled={!searchOptions.sort.salary.status}
onClick={() => {
setSearchOptions({
...searchOptions,
sort: {
...searchOptions.sort,
salary: {
...searchOptions.sort.salary,
desc: !searchOptions.sort.salary.desc,
},
},
});
}}
>
{searchOptions.sort.salary.desc ? (
<ArrowDownwardIcon />
) : (
<ArrowUpwardIcon />
)}
</IconButton>
</Grid>
</Grid>
<Grid
item
```

```
container
xs={4}
justify="space-around"
alignItems="center"
style={{ border: "1px solid #D1D1D1", borderRadius: "5px" }}
>
<Grid item>
<Checkbox
name="duration"
checked={searchOptions.sort.duration.status}
onChange={(event) =>
setSearchOptions({
...searchOptions,
sort: {
...searchOptions.sort,
duration: {
...searchOptions.sort.duration,
status: event.target.checked,
},
},
})
}
id="duration"
/>
</Grid>
<Grid item>
<label for="duration">
<Typography>Duration</Typography>
</label>
</Grid>
<Grid item>
<IconButton
disabled={!searchOptions.sort.duration.status}
onClick={() => {
setSearchOptions({
...searchOptions,
sort: {
...searchOptions.sort,
duration: {
...searchOptions.sort.duration,
desc: !searchOptions.sort.duration.desc,
},
},
});
```

```
}}
>
{searchOptions.sort.duration.desc ? (
<ArrowDownwardIcon />
) : (
<ArrowUpwardIcon />
)}
</IconButton>
</Grid>
</Grid>
<Grid
item
container
xs={4}
justify="space-around"
alignItems="center"
style={{ border: "1px solid #D1D1D1", borderRadius: "5px" }}
>
<Grid item>
<Checkbox
name="rating"
checked={searchOptions.sort.rating.status}
onChange={(event) =>
setSearchOptions({
...searchOptions,
sort: {
...searchOptions.sort,
rating: {
...searchOptions.sort.rating,
status: event.target.checked,
},
},
})
}
id="rating"
/>
</Grid>
<Grid item>
<label for="rating">
<Typography>Rating</Typography>
</label>
</Grid>
<Grid item>
<IconButton
```

```
disabled={!searchOptions.sort.rating.status}
onClick={() => {
setSearchOptions({
...searchOptions,
sort: {
...searchOptions.sort,
rating: {
...searchOptions.sort.rating,
desc: !searchOptions.sort.rating.desc,
},
},
});
}}
>
{searchOptions.sort.rating.desc ? (
<ArrowDownwardIcon />
) : (
<ArrowUpwardIcon />
)}
</IconButton>
</Grid>
</Grid>
</Grid>
</Grid>

<Grid item>
<Button
variant="contained"
color="primary"
style={{ padding: "10px 50px" }}
onClick={() => getData()}
>
Apply
</Button>
</Grid>
</Grid>
</Paper>
</Modal>
);
};

const Home = (props) => {
const [jobs, setJobs] = useState([]);
const [filterOpen, setFilterOpen] = useState(false);
```

```
const [searchOptions, setSearchOptions] = useState({
query: "",
jobType: {
fullTime: false,
partTime: false,
wfh: false,
},
salary: [0, 100],
duration: "0",
sort: {
salary: {
status: false,
desc: false,
},
duration: {
status: false,
desc: false,
},
rating: {
status: false,
desc: false,
},
},
});

const setPopup = useContext(SetPopupContext);
useEffect(() => {
getData();
}, []);

const getData = () => {
let searchParams = [];
if (searchOptions.query !== "") {
searchParams = [...searchParams, `q=${searchOptions.query}`];
}
if (searchOptions.jobType.fullTime) {
searchParams = [...searchParams, `jobType=Full%20Time`];
}
if (searchOptions.jobType.partTime) {
searchParams = [...searchParams, `jobType=Part%20Time`];
}
if (searchOptions.jobType.wfh) {
searchParams = [...searchParams, `jobType=Work%20From%20Home`];
}
```

```
if (searchOptions.salary[0] != 0) {
searchParams = [
...searchParams,
`salaryMin=${searchOptions.salary[0] * 1000}`,
];
}
if (searchOptions.salary[1] != 100) {
searchParams = [
...searchParams,
`salaryMax=${searchOptions.salary[1] * 1000}`,
];
}
if (searchOptions.duration != "0") {
searchParams = [...searchParams, `duration=${searchOptions.duration}`];
}

let asc = [],
desc = [];

Object.keys(searchOptions.sort).forEach((obj) => {
const item = searchOptions.sort[obj];
if (item.status) {
if (item.desc) {
desc = [...desc, `desc=${obj}`];
} else {
asc = [...asc, `asc=${obj}`];
}
}
});
searchParams = [...searchParams, ...asc, ...desc];
const queryString = searchParams.join("&");
console.log(queryString);
let address = apiList.jobs;
if (queryString !== "") {
address = `${address}?${queryString}`;
}

axios
.get(address, {
headers: {
Authorization: `Bearer ${localStorage.getItem("token")}`,
},
})
.then((response) => {
```

```
console.log(response.data);
setJobs(
response.data.filter((obj) => {
const today = new Date();
const deadline = new Date(obj.deadline);
return deadline > today;
})
);
})
.catch((err) => {
console.log(err.response.data);
setPopup({
open: true,
severity: "error",
message: "Error",
});
});
};

return (
<>
<Grid
container
item
direction="column"
alignItems="center"
style={{ padding: "30px", minHeight: "93vh" }}
>
<Grid
item
container
direction="column"
justify="center"
alignItems="center"
>
<Grid item xs>
<Typography variant="h2">Jobs</Typography>
</Grid>
<Grid item xs>
<TextField
label="Search Jobs"
value={searchOptions.query}
onChange={(event) =>
setSearchOptions({
```

```
...searchOptions,
query: event.target.value,
})
}
onKeyPress={(ev) => {
if (ev.key === "Enter") {
getData();
}
}}
InputProps={{
endAdornment: (
<InputAdornment>
<IconButton onClick={() => getData()}>
<SearchIcon />
</IconButton>
</InputAdornment>
),
}}
style={{ width: "500px" }}
variant="outlined"
/>
</Grid>
<Grid item>
<IconButton onClick={() => setFilterOpen(true)}>
<FilterListIcon />
</IconButton>
</Grid>
</Grid>

<Grid
container
item
xs
direction="column"
alignItems="stretch"
justify="center"
>
{jobs.length > 0 ? (
jobs.map((job) => {
return <JobTile job={job} />;
})
)
:
(
```

```
<Typography variant="h5" style={{ textAlign: "center" }}>
No jobs found
</Typography>
)}
</Grid>
{/* <Grid item>
<Pagination count={10} color="primary" />
</Grid> */}
</Grid>


<FilterPopup
open={filterOpen}
searchOptions={searchOptions}
setSearchOptions={setSearchOptions}
handleClose={() => setFilterOpen(false)}
getData={() => {
getData();
setFilterOpen(false);
}}
/>
</>
);
};
export default Home;
```

6 Conclusions

This research has focused on improving the education environment by developing a knowledge sharing system that acts as a job portal. A job web portal provides an efficient search for online information on job vacancies for jobseekers. The main goal of this portal is to attempt to produce the right graduates based on the industry needs. However, it is important to be aware the job web portals can never fulfill all the problems of jobless graduates.

It can be concluded that this project of Online Job Portal was a real learning experience. The principles of software production were well implemented throughout the system. The project has been made as per as the given specifications. The Online Job Portal developed by us is purely based on MERN Stack platform.

7 Acknowledgments

I would like to extend my gratitude to my supervisor, Dr. Rossitza Marinova, whose guidance and support made this research possible. I am also very grateful to all my professors who helped and taught me during my graduate program. A good support system is important to surviving and staying sane in graduate school. Thanks to my family, friends, and classmates for sticking with me through this graduate program and throughout my endeavors.

This work is a part of the Final Year Project of Concordia University of Edmonton, Alberta, Department of Mathematical and Physical Sciences Faculty of Graduate Studies.

8 Privacy Policy

We are committed to maintaining the accuracy, confidentiality, and security of your personally identifiable information ("Personal Information"). As part of this commitment, our privacy policy governs our actions as they relate to the collection, use and disclosure of Personal Information.

1. Identifying Purposes

We collect, use and disclose Personal Information to provide you with the product or service you have requested and to offer you additional products and services we believe you might be interested in. The purposes for which we collect Personal Information will be identified before or at the time we collect the information. In certain circumstances, the purposes for which information is collected may be clear, and consent may be implied, such as where your name, address and payment information is provided as part of the order process.

2. Consent

Knowledge and consent are required for the collection, use or disclosure of Personal Information except where required or permitted by law. Providing us with your Personal Information is always your choice. However, your decision not to provide certain information may limit our ability to provide you with our products or services. We will not require you to consent to the collection, use, or disclosure of information as a condition to the supply of a product or service, except as required to be able to supply the product or service.

3. Limiting Collection

The Personal Information collected will be limited to those details necessary for the purposes identified by us. With your consent, we may collect Personal Information from you in person, over the telephone or by corresponding with you via mail, facsimile, or the Internet.

4. Limiting Use, Disclosure and Retention

Personal Information may only be used or disclosed for the purpose for which it was collected unless you have otherwise consented, or when it is required or permitted by law. Personal Information will only be retained for the period of time required to fulfill the purpose for which we collected it or as may be required by law. [If applicable, include a description of any parties with whom you may share Personal Information.]

5. Safeguarding Customer Information

Personal Information will be protected by security safeguards that are appropriate to the sensitivity level of the information. We take all reasonable precautions to protect your Personal Information from any loss or unauthorized use, access or disclosure.

9 Terms and Conditions

Please read these terms and conditions ("terms and conditions", "terms") carefully before using In-House Job Portal website.

Conditions of use

By using this website, you certify that you have read and reviewed this Agreement and that you agree to comply with its terms. If you do not want to be bound by the terms of this Agreement, you are advised to leave the website accordingly. We only grants use and access of this website, its products, and its services to those who have accepted its terms.

1. Privacy policy

   Before you continue using our website, we advise you to read our privacy policy regarding our user data collection. It will help you better understand our practices.

2. User accounts

   As a user of this website, you may be asked to register with us and provide private information. You are responsible for ensuring the accuracy of this information, and you are responsible for maintaining the safety and security of your identifying information. You are also responsible for all activities that occur under your account or password. If you think there are any possible issues regarding the security of your account on the website, inform us immediately so we may address them accordingly.

   We reserve all rights to terminate accounts, edit or remove content and cancel orders at our sole discretion.

References

[1] Dorn, Jrgen and Naz, Tabbasum and Pichlmair (2007) Ontology development for job portal resource management, Markus, Knowledge Management: Innovation, Technology and Cultures , 109–120.

[2] Mansourvar, M., Yasin, N. B. M. (2014). Development of a job web portal to improve education quality. International Journal of Computer Theory and Engineering, 6(1), 43.

[3] Pinjari, M., De, N., Kokne, R., Siddiqui, A., Chitre, D. (2019). Online Job Portal. International Research Journal of Engineering and Technology.

[4] Pardamean, Bens. "Users' interest assessment on job portal." International Journal of Web Portals (IJWP) 6, no. 1 (2014): 64-75.

[5] Khan, Muhammad Sabeeh, and Khan, Muhammad Shaff. Online job portal. Diss. University of Management and Technology Lahore, 2015.

[6] ICAT Job Portal: a generic job submission system built on a scientific data catalog. Fisher, Stephen M and Phipps, Kevin and Rolfe, Daniel, 2013.

[7] Mehra, M., Kumar, M., Maurya, A., Sharma, C. (2021). MERN stack Web Development. Annals of the Romanian Society for Cell Biology, 25(6), 11756-11761.

[8] Abubucker Samsudeen Shaffi and Mohaned Al-Obaidy, "Analysis and comparative study of traditional and web information systems development methodology (WISDM) towards web development applications," International Journal of Emerging Technology and Advanced Engineering, Volume 3, Issue 11, November 2013.

[9] Pooja T. Killewale and Prof. A.R. Mune, "Job Portal – A web application for distributed clients," International Journal of Advanced Research in Computer and Communication Engineering, Vol. 6, Issue 5, May 2017.

[10] Beblavý, M., Kureková, L. M., Haita, C. (2016). The surprisingly exclusive nature of medium-and low-skilled jobs: Evidence from a Slovak job portal. Personnel Review.

[11] Jason Crawford.
`https://www.websitepolicies.com/blog/sample-terms-conditions-template`