ONLINE JOB NOTIFIER

A Main Project report submitted

in partial fulfilment of requirements

for the award of degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

SRILEKHA DEVINEEDI	**(Reg No:19131A05M7)**
USHA LALAM	**(Reg No:19131A05P0)**
SURYA CHIKKALA	**(RegNo:19131A05N2)**
CHITNEEDI SRIDEVI	**(Reg No:19131A0545)**

Under the esteemed guidance of

Dr. N.V.S LAKSHMIPATHI RAJU

Associate Professor

Department of Computer Science and Engineering

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING

(AUTONOMOUS)

(Affiliated to JNTU-K, Kakinada, AP)

VISAKHAPATNAM-530048

2022 – 2023

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING

(AUTONOMOUS)

VISAKHAPATNAM-530048

CERTIFICATE

This is to certify that the main project report entitled

"Online Job Notifier" being submitted

by

SRILEKHA DEVINEEDI	**(Reg No:19131A05M7)**
USHA LALAM	**(Reg No:19131A05P0)**
SURYA CHIKKALA	**(RegNo:19131A05N2)**
CHITNEEDI SRIDEVI	**(RegNo:19131A0545)**

in their VIII semester in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology

in

Computer Science and Engineering

during the academic year 2022-2023

Signature of the Guide

Dr. N.V.S LAKSHMIPATHI RAJU

Associate Professor
Computer Science and Engineering
GVPCE(A)

Signature of the HOD

Dr. D.N.D. HARINI

Associate Professor & HOD
Computer science and engineering
GVPCE(A)

Project Viva-voice held on _____

External Examiner

DECLARATION

We hereby declare that this main project entitled **"ONLINE JOB NOTIFIER"** is a bonafide work done by us and submitted to the **Department of Computer Science and Engineering, Gayatri Vidhya parishad college of engineering (Autonomous) Visakhapatnam,** in partial fulfilment for the award of the degree of B.Tech is of our own and it is not submitted to any other university or has been published any time before.

PLACE: VISAKHAPATNAM
DATE:

SRILEKHA .D (RegNo:19131A05M7)
USHA .L (RegNo:19131A05P0)
SURYA .C (Reg No:19131A05N2)
C. SRIDEVI (RegNo:19131A0545)

ACKNOWLEDGEMENT

We would like to express our deep sense of gratitude to our esteemed institute **Gayatri Vidya Parishad College of Engineering (Autonomous),** which has provided us an opportunity to fulfil our cherished desire.

We express our sincere thanks to our Principal **Prof. Dr. A. B. KOTESWARARAO, Gayatri Vidya Parishad College of Engineering (Autonomous)**for his encouragement to us during the course of this project, giving us a chance to explore and learn new technologies in the form of main project.

We express our sincere thanks to **Dr. D.N.D. HARINI, Associate Professor and Head of the Department of Computer Science and Engineering, Gayatri Vidya Parishad College of Engineering (Autonomous),** for giving us an opportunity to do the project in college.

We express our profound gratitude and our deep indebtedness to our guide **Dr. N.V.S LAKSHMIPATHI RAJU, Associate Professor, Department of Computer Science and Engineering,** whose valuable suggestions, guidance and comprehensive assistance helped us a lot in realizing our present project.

We also thank our coordinator, **Dr. C.H. SITA KUMARI, Associate Professor,** and **Project Coordinator, Department of computer science and engineering,** for the kind suggestions and guidance for the successful completion of our project work.

Finally we would also like to thank all the members of the teaching and non-teaching staff of the Computer Science and Technology Department for all their support in completion of our project.

SRILEKHA DEVINEEDI	**(Regd.No:19131A105M7)**
USHA LALAM	**(Regd.No:19131A105P0)**
SURYA CHIKKALA	**(Regd.No:19131A105N2)**
CHITNEEDI SRIDEVI	**(Regd.No:19131A0545)**

ABSTRACT

Internet is a source of live data that is constantly updating with data of almost any field. Having tools that can automatically detect these updates and select that information that we are interested in, will have utmost importance now-a-days. Web scraping will help to do it. Web scraping is a method of extracting and restricting the data from web pages.

This application will find new openings on job portals and alert the user by providing required information.

It will reduce the manual work of going to a website and searching for new job multiple times. As the application collects the data from job portal and send it to the user.

Keywords:

- web scrapping, job notifications, jobseekers, applicants, employers, recruiter, awareness.

CONTENTS

1. INTRODUCTION

Web Scraping is a web technique for extracting data from the web and turning unstructured data on the web (including HTML formats) into structured data that you can store to your local computer or database. The web-based application is basically concerned with different job services provided by different companies. It is also concerned with details of jobseekers. Jobseeker can view the list of different jobs and can apply for jobs.[1]

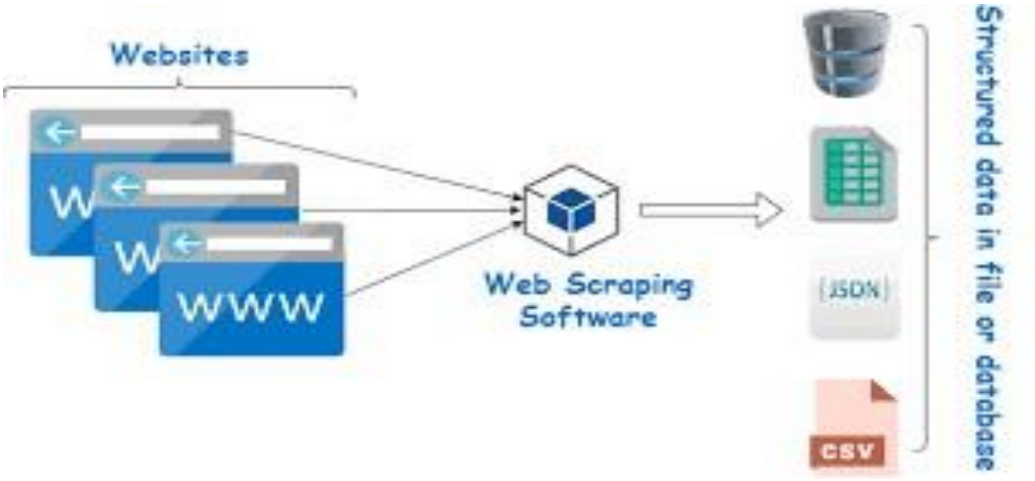


Fig 1.1 web scrapping software

1.1 OBJECTIVE:

The main objective of the application is to find new openings on job portals and alert the user by providing required information.

The online job notifier system that is to be developed provides the members with jobs information, online applying for jobs and many other facilities. This system provides service to the job applicants to search for working opportunities. This Project will primarily focus on the posting and management of job vacancies by notifying the users. This system is designed such that ultimately all vacancies will be posted online and would offer employers the facilities to post their vacancies online. It helps to review and manage the resulting applications efficiently through the web.

1.2 PURPOSE:

The old system requires applicants to search through print and visual media for job opportunities. Applicants need to apply for jobs using conventional methods and appear for interview on a specific date at a specified location. Employers need to advertise the vacancies and sort all applicant details, conduct selection procedures and complete the formalities. This approach is tedious and requires much effort and resources. So, to overcome this the solution to the problem is “online job notifier” where jobseekers easily find the jobs and employer can find suitable candidates for the job.[2]

1.3 SCOPE:

This application can be used on different dynamic websites i.e.., job portals and automate the repetitive tasks. It will reduce the manual work of going to a website and searching for new job multiple times. As the application collects the data from job portal and send it to the user.

The online job notifier that is developed provide jobseekers with different jobs information like:

- Online applying for jobs
- Search for jobs
- Applying for jobs

Supported by well-designed database and web-based design.

The basic scope contains:

- Jobseeker’s Area
- Company’s Area
- Administrator’s Area

2. SRS DOCUMENT

A software requirements specification (SRS) is a document that describes what the software will do and how it will be expected to perform. It also describes the functionality the product needs to fulfill all stakeholders (business, users) needs.

2.1 FUNCTIONAL REQUIREMENTS

A functional requirement defines a function of a system or its component, where a function of a system or its component is described as a specification of behaviour between inputs and outputs.
The functional requirements in our project are:

MODULES:

- User module
- Company module

INPUT:

User should provide:

Job Type-Job title, keywords, or company

Location-City, state, or pin code

PROCESS:

- Scraping the data from job portal and transforming the data.
- Alerting the user by sending the required information.

OUTPUT: New Job Openings are sent to user.

2.2 NON-FUNCTIONAL REQUIREMENTS

A Non-Functional Requirement (NFR) defines the quality attribute of a software system. They judge the software system based on Security, Performance, Maintainability, Reliabilty, Security, Safety and other non-functional standards that are critical to success of the software system.

Non-Functional Requirements are:

Performance:

Response Time: The system provides acknowledgment in just a few seconds.

Capacity: The system needs to support at least 100 people at once.

User-Interface: The user interface acknowledges within five seconds.

Maintainability:

Back-Up: The system offers the efficiency for data backup.

Reliability:
Availability: The system is available all the time.

Cache:

We can cache URLs that are frequently accessed by the users. The UGS servers, before making a query to the database, may check if the cache has the desired URL. Then it does not need to makethe query again**.**

Security:

The job portal will provide restriction against unauthorized access.

Safety:

There will be backup of data for any future mishap.

2.3 SOFTWARE REQUIREMENTS

Software requirements deal with defining software resource requirements that need to be installed on a computer to provide optimal functioning of an application.

OPERATING SYSTEM: Windows7/XP/8/10

TECHNOLOGIES USED: JAVASCRIPT, NODE JS

IDE: VS CODE

2.4 HARDWARE REQUIREMENTS

Processor: Intel dual core i3/i5/i7

RAM: 4 GB

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Web scraping technique is used that helps to collect the data (job openings data) and transferring either, CSV or json file to help you better understand the information you have gathered.

The old system requires applicants to search through print and visual media for job opportunities. Applicants need to apply for jobs using conventional methods and appear for interview on a specific date at a specified location. Employers need to advertise the vacancies and sort all applicant details, conduct selection procedures and complete the formalities. This approach is tedious and requires much effort and resources.[3]

DRAWBACKS:

- The most and primitive drawback of the existing work is that it requires a lot of the user's time and manual work for selection.
- Any platform right now synchronizes the user's responses and shows the list of all the options that are in the database but unlike this our project performs all the analysis and sends a personalized selection of the jobs to the user's socials.

3.2 PROPOSED SYSTEM

This application will automate the repetitive tasks.

It will reduce the manual work of going to a website and searching for a new job-openings multiple time.

Simple and professional GUI users of all qualification groups. Increased filtering for employees seeking job as a fresher or as an experienced individual.

Adding notifier and periodic scheduling that will alert the user when the job applications are open after specified interval of time.

3.3 PROJECT FEASIBILITY STUDY

Feasibility study includes consideration of all the possible ways to provide a solution to the given problem. The proposed solution should satisfy all the user requirements and should be flexible enough so that future changes can be easily done based on the future upcoming requirements.

3.3.1 ECONOMICAL FEASIBILITY

Economical feasibility is the measure to determine the cost and benefit of the proposed system. A project is economical feasible which is under the estimated cost for its development. These benefit and costs may be tangible or intangible. Job Portal is the cost-effective project in which there is less possibility of intangible cost so there is no difficulty to determine the cost of the project.

3.3.2 TECHNICAL FEASIBILITY

Technical feasibility study is concerned with specifying equipment and software that will successfully satisfy the user requirement; the technical needs of the system may vary considerably. The facility to produce outputs in a given time. Our project is a web based application which is based on client-server based application. In this application every page as output is render from server to client so it is necessary that the page should be rendered in time. For this I have avoided more and more code in the page- load event.

3.3.3 OPERATIONAL FEASIBILITY

Operation feasibility is used to check whether the project is operationally feasible or not. Our project is mainly different from the other system because of its web-support feature. So, the measure for operational feasibility is something different from other system. Generally, the operational feasibility is related to organization aspects. The change determination is as such that early product were either a man or group of men or the jobs based manual but now a day with the advent of Internet technology.

4. SOFTWARE DESCRIPTION

4.1 JAVASCRIPT

JavaScript often abbreviated as **JS**, is a programming language that conforms to the ECMAScript specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object orientation, and first-class functions. As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative programming styles.

4.2 REACT

ReactJS is a **declarative**, **efficient**, and flexible **JavaScript library** for building reusable UI components. It is an open-source, component-based front end library which is responsible only for the view layer of the application. The main objective of ReactJS is to develop User Interfaces (UI) that improves the speed of the apps. It uses virtual DOM (JavaScript object), which improves the performance of the app.

4.3 Node.js

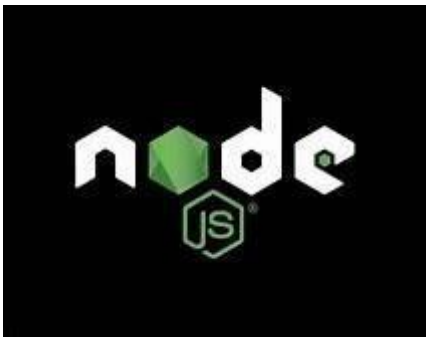


Fig 4.3 Node.js

As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications.**Node.js** is an open-source, cross-platform, back-end

JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser.

4.4 SELENIUM WEB DRIVER

Selenium WebDriver is a library that allows the controlling of web browsers automatically. WebDriver is the most widely used automation tool for web applications and supports all major browsers, including Chrome, Firefox, Internet Explorer and Edge.[6]

It comprises of four major components which include:

- Selenium Integrated Development Environment (IDE)
- Selenium Remote Control (Now Deprecated)
- WebDriver
- Selenium Grid

4.5 CHEERIO

Cheerio is a tool for parsing HTML and XML in Node.js, It is fast, flexible, and easy to use. Cheerio is an implementation of jQuery that works on a virtual DOM. The DOM is built from an HTML string without running any JavaScript or applying CSS styles. Since it's not displaying anything, this makes it a great way to scrape data on a server, or if you're creating a service hosted by a cloud provider, you can run it in a serverless function.[7]

4.6 NODE-CRON

Cron-job can be used to schedule logging tasks and monitor server resources in our Node.js applications. Let's say something happened, like a network delay or a Warning message. We can schedule a cron-job to log at a specific time or interval to track our server status. cron is a utility program that lets users input commands for scheduling tasks repeatedly at a specific time. Tasks scheduled in cron are called cron-jobs Users can determine what kind of task they want to automate and when it should be executed. Cron is a daemon – a background process executing non-interactive jobs.[4]

4.7 SHELL JS

A shell command is one that is processed internally by the shell. There is no corresponding executable program. Take cd for instance. There is no /bin/cd program, say, and which cd specifies that it is a built-in command. It is part of the shell. Node.js can run shell commands by using the standard child_ process module.

4.8 NODEMAILER

Nodemailer is a Node.js module that allows you to send emails from your server with ease. Whether you want to communicate with your users or just notify yourself when something has gone wrong, one of the options for doing so is through mail.

NODE MAILER FEATURES

- A single module with zero dependencies – code is easily auditable, as there are no dark corners
- Unicode support to use any characters, including emoji
- Use HTML content, as well as plain text alternative

5. PROJECT DESCRIPTION

5.1 PROBLEM DEFINITION

To find new openings on job portals and alert the user by providing required information.

The old system requires applicants to search through print and visual media for job opportunities. Applicants need to apply for jobs using conventional methods and appear for interview on a specific date at a specified location. Employers need to advertise the vacancies and sort all applicant details, conduct selection procedures and complete the formalities. This approach is tedious and requires much effort and resources.

The old system requires applicants to search through print and visual media for job opportunities. Applicants need to apply for jobs using conventional methods and appear for interview on a specific date at a specified location. Employers need to advertise the vacancies and sort all applicant details, conduct selection procedures and complete the formalities. This approach is tedious and requires much effort and resources.

5.2 PROJECT OVERVIEW

- User specifies the job type and location
- Scrap the data from the job portal and transform it structured data.
- This data is sent to the user using Gmail and this process is repeated continuously after a specified amount of time.
- Having tools that can automatically detect these updates and select that information that we are interested in, will have utmost importance now-a-days. Web scraping will help to do it. Web scraping is a method of extracting and restricting the data from web pages.

This application will find new openings on job portals and alert the user by providing required information. It will reduce the manual work of going to a website and searching for new job multiple times. As the application collects the data from job portal and send it to the user.

5.3 MODULE DESCRIPTION

WEB SCRAPPING:

Web Scrapping is a web technique for extracting data from the web and turning unstructured data on the web (including HTML formats) into structured data that you can store to your local computer or a database.[5]

- One way is to manually copy-paste the data, which both tedious and time-consuming.
- Web Scraping is the automation of the data extraction process from websites.
- This event is done with the help of web scraping software known as web scrapers.
- They automatically load and extract data from the websites based on user requirements. These can be custom built to work for one site or can be configured to work with any website.

6. SYSTEM DESIGN

6.1 SYSTEM ARCHITECTURE

A system architecture is a set of principles that define the way software is designed and developed.[8]

In this architecture web crawling process is processed where it is the process of indexing data on web page by using program automated script. In this online application web crawling functions as follows: systematically browse the web to index content for search engines. Web crawling copy pages for processing by a search engine, which indexes the downloaded pages for easier retrieval so that users can get search results faster. Web crawling helps to automatically retrieving

content from any web page.

Now the html script from the website that have specific patterns are to be retrieved in the process of web crawling, using this source pentaho can retrieve data from the website in which pentaho provides data integration, data mining and extract from which data will be successfully obtained and the whole data can be formed as an excel sheet id the data is not obtained then again the process will repeat from extracting data in search engines.

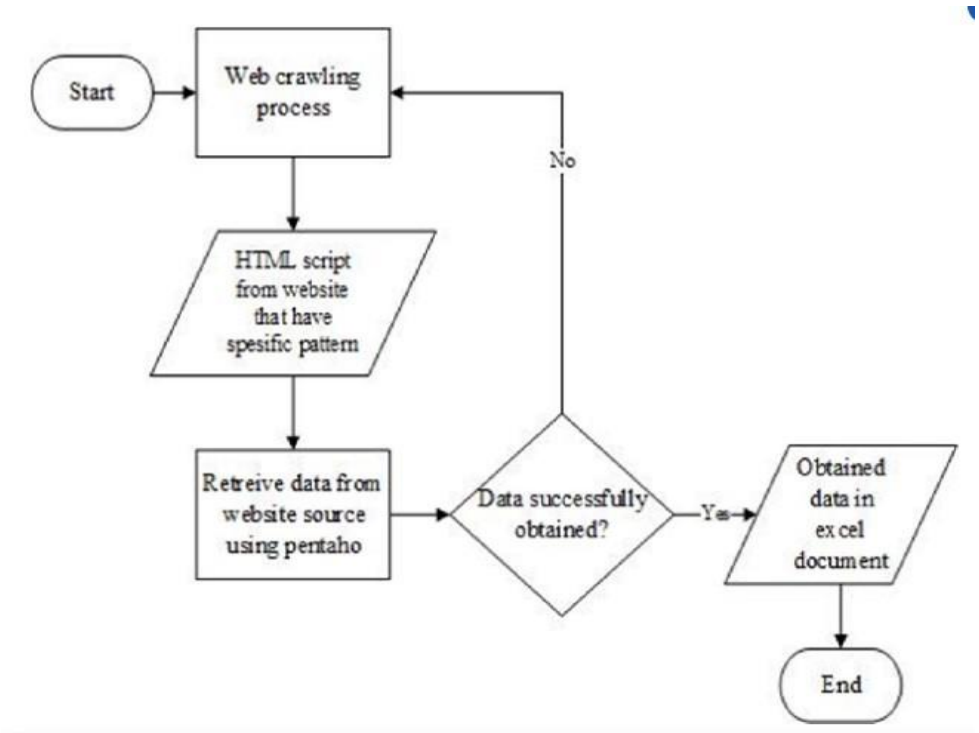


Fig 6.1 System Architecture

6.2 ACTIVITY DIAGRAM

In the above activity diagram the execution process is displayed as a flow chart diagram where the user opens the website which is redirecting from the frontend code that is developed then the user will enter the details i.e., the user's email, preferred location and the job role according to the skill level after submitting the code will be redirecting to the submit page. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The control flow is drawn from one operation to another. It captures the dynamic behaviour of the system.

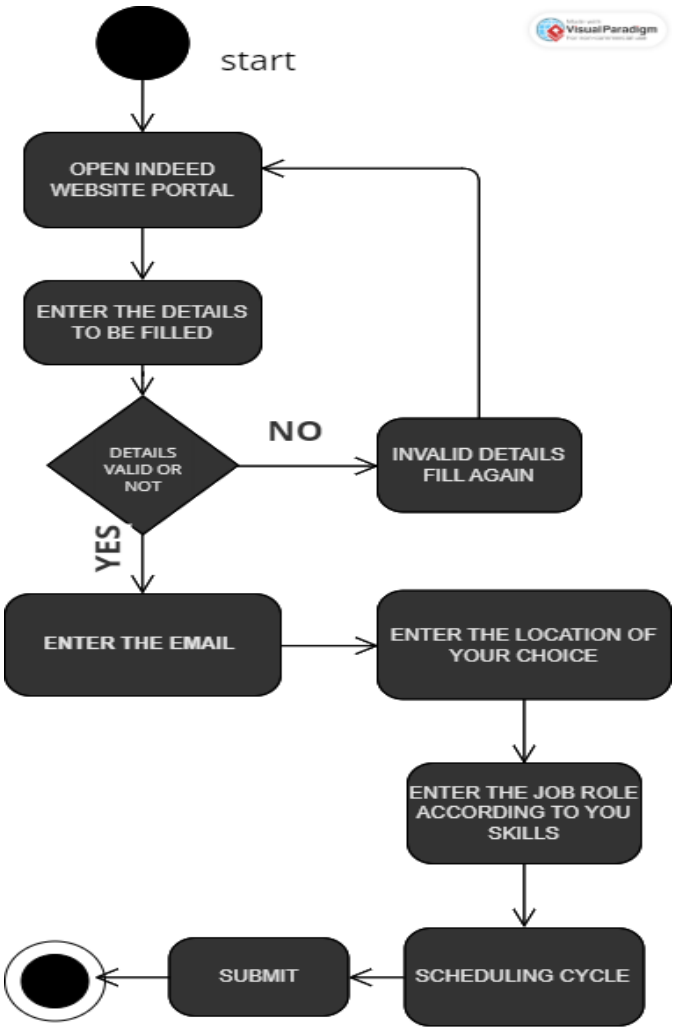


Fig 6.2 Activity Diagram

6.3 STATE CHART DIAGRAM

A **state chart diagram** shows a state machine, consisting of states, transitions, events, and activities. State chart diagrams address the dynamic view of a system.[9]

The system works accordingly, the process done in the job portal is as follows, the unstructured data is extracted based on user requirements using cheerio and selenium driver packages, the required information from the unstructured data is parsed and formatted using the packages imported in the code which gives the output as structured data providing user email credentials, those credentials are sent to the server and when updated the user will be notified and this process will repeat after every specified interval of time using cronjob package.

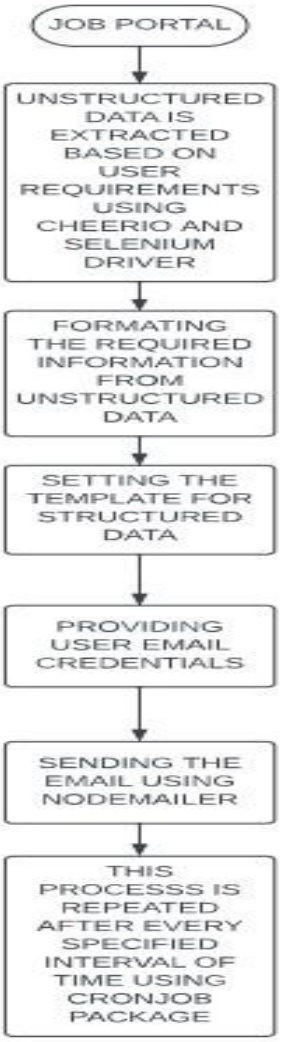


Fig 6.3 State Chart Diagram

6.4 USE CASE DIAGRAM

The use case diagram is portraying the dynamic aspect of the system.

Here the user actor makes use of the website for searching and viewing jobs in the indeed portal from the information sent by the company actor regarding the job posts/job vacancies. Company actor will upload the jobs and the server or the admin will approve the jobs i.e., managing the job posts to select the jobseekers and the company actor will download the details and update the details of the user in the database to notify the user about the job posts. The user can set the scheduling time on the day and time when they are free. So as per the time they have given, they will be receiving updates every week.

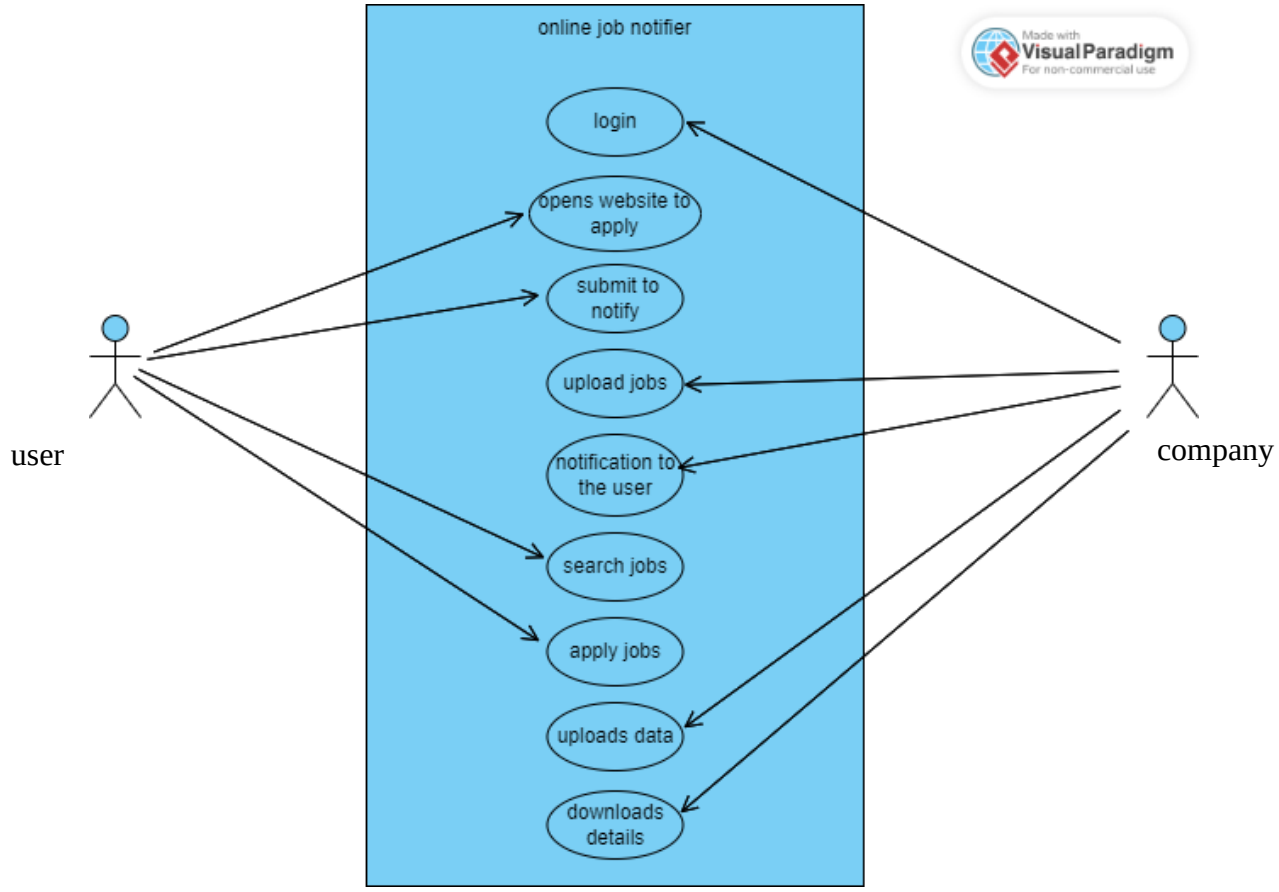


Fig 6.4 Use-Case Diagram

6.5 CLASS DIAGRAM

Class diagram represents the static view of the application . It describes attributes and operations of a class.

Here the class job_notifier_website is the main class in which it contains the attributes enter email, enter location, enter job role, scheduling time and the operations mentiones are email notification, jobpostsactivity, jobvacancies, filtering location and job roles in which four of them are performing actions. In class diagram there will be relation between one class to another class like onetomany(1..n), onetoone(1…1), zero to many(0…*), the sever here act as an important role in which the details of the user credentials are uploaded to the database through which the user will be notified about the vacancies according to their scheduled time given by the user.

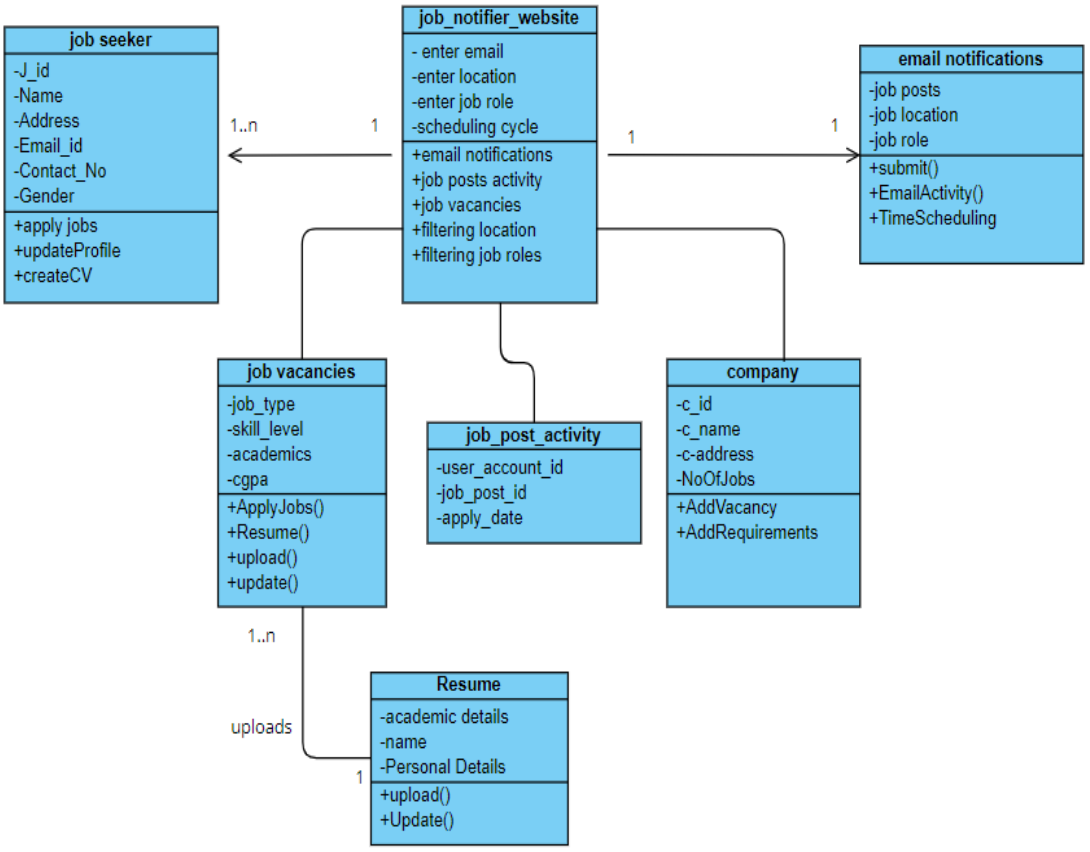


Fig 6.5 Class Diagram

7. DEVELOPMENT

7.1 SAMPLE CODE

BACKEND CODE

7.1.1 app.js

//importing all the dependencies required for the project

```
const webdriver = require("selenium-webdriver");
const cheerio = require("cheerio");
const cron = require("node-cron");
let shell = require("shelljs");
const nodemailer = require("nodemailer");
```

//To automate the process in the chrome

```
const driver = new webdriver.Builder().forBrowser("chrome").build();

const express = require("express");
const app = express();
const port = 4000;
const cors = require("cors");
const a = require("cors");

app.use(express.urlencoded({extended:true}))
app.use(express.json())
app.use(cors())

app.get("/", cors(), async(req,res)=>{
   res.send("This is working");
})
```

//Retreiving the inputs given by the user in the frontend

```
app.post("/post_name",async(req,res) => {
   let {e} = req.body
   let {l} = req.body
   let {j} = req.body
   let {t} = req.body
   let {d} = req.body
   console.log(e,l,j,t,d);
  const arr = t.split(":");
  if (arr[1]==="00"){
   arr[1]="0";
  }
```

```
 const r = {
   sunday: 0,
   monday : 1,
   tuesday: 2,
   wednesday : 3,
   thursday:4,
   friday:5,
   saturday:6,
 }
```

//concatinating the string to pass the input for cron job scheduler

```
var s = arr[1] + " " + arr[0] + " * * * " + r[d];
```

//The async-await function waits until the data is received and as soon as the data is received,the process gets started.

```
async function getJobs(jobQuery, place) {
 try {
   const caseUrl = "https://in.indeed.com/jobs";
```

//The base url is updates dynamically by the user inputs

```
   const url = caseUrl + "?q=" + jobQuery + "&l=" + place + "&from=searchOnHP";
   console.log("Url:", url);
```

//After posting get request for the url,it waits till it receives a response,and then continues the procedure

```
   await driver.get(url);
```

//The data from the indeed website in the form of html and it is unstructured.

```
   const html = await driver.getPageSource();
```

//Parsing the unstructured data

```
   const $ = cheerio.load(html);
```

//Specifying the attributes that must be present in the email
//rawElements is an array of objects extracting all the details of jobs

```
   const rawElements = {
     title: [...$(".jobTitle > a > span").contents()].map((el) => el.data),
     company: [...$("span.companyName").contents()].map((el) => el.data),
     location: [...$(".companyLocation").contents()].map((el) => el.data),
     description: [...$("div.job-snippet")].map((e) => cheerio.text($(e)).trim()),
```

```
    links: [...$(".jobTitle > a")].map(
      (a) => "https://in.indeed.com" + a.attribs.href
    ),
   };
```

//printing the jobs that are stored with the help of map function

```
   const jobs = rawElements.title.map((_, i) => ({
    title: rawElements.title[i],
    company: rawElements.company[i],
    location: rawElements.location[i],
    description: (rawElements.description[i] || "").trim(),
    link: (rawElements.links[i] || "").trim(),
   }));
   console.log({ jobs });
   return jobs;
  }
  finally {
   driver.quit();
  }
}

async function task() {
  let retrivedJobs = await getJobs(
   j,  /role/
   l /location/
  );
  console.log({ retrivedJobs });
```

//Specifying the template of how the user get mails.

```
  const liEles = retrivedJobs.map(
   ({ title, company, location, description, link }) => `
        Title - ${title}
        Company - ${company}
        Location - ${location}
        Description-${description}
        Link-${link}
   );
```

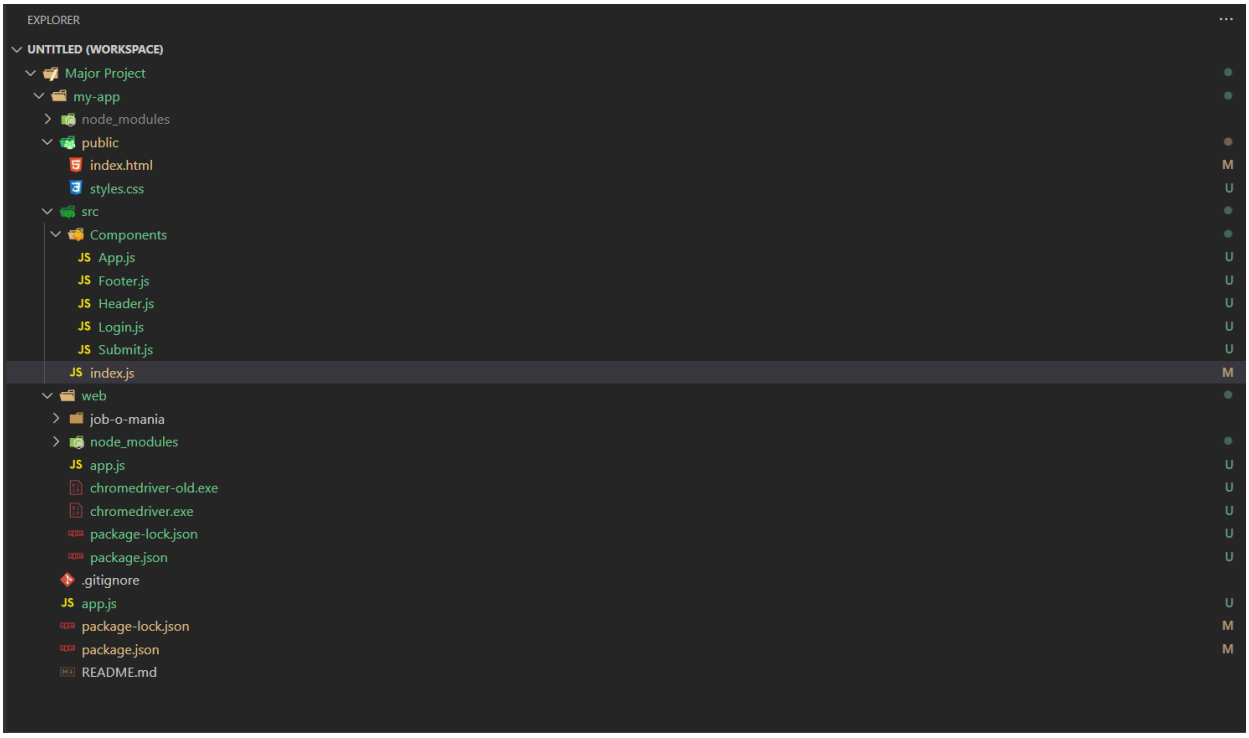
//Giving the credentials

```
  const transporter = nodemailer.createTransport({
   service: "hotmail",
   auth: {
    user: "ajaykarthik633@outlook.com",
    pass: "Ajay@1234",
   },
```

```
  });

  console.log(e);

  //Specifying what to send and whom to send

  const options = {
    from: "ajaykarthik633@outlook.com",
    to: e,  /email/
    subject: "Your hunt for new jobs!",
    text: liEles.join("\n"),
  };
  transporter.sendMail(options, function (err, info) {
    if (err) {
      console.log(err);
      return;
    }
    console.log("Sent: " + info.response);
  });
}
task();

//Scheduling the mail updates by the time specified by the user

cron.schedule(s,function(){
    console.log("Schedule is running..");
    if(shell.exec(JSON.stringify(getJobs(j, l))).code!==0){
        console.log("something went wrong");
      task()
}
});

})

//Listening at the port number mentioned in the frontend axios.

app.listen(port, () =>{
  console.log('Listening at http://localhost:4000');
})
```

FRONTEND CODE

React-app files structure:



7.1.2 index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>React App</title>
    <link rel="stylesheet" href="styles.css"/>
  </head>
  <body>
    <script src="./src/index.js"></script>
    <div id="root"></div>
  </body>
</html>
```

7.1.3 index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import { BrowserRouter } from 'react-router-dom';
// import './index.css';
import App from './Components/App';
// import reportWebVitals from './reportWebVitals';
```

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <BrowserRouter>
       <App />
    </BrowserRouter>
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
// reportWebVitals();
```

7.1.4 App.js

```
import React from "react";
import Login from "./Login"
import Header from "./Header";
import Footer from "./Footer"
import Submit from "./Submit";
import { Routes,Route } from "react-router-dom";

function App(){
   return(
      <div>
{/* Separating the code into simplier react components and mentioning the route
so that when user
      clicks the submit button it renders to other page */}
         <Header/>
         <Routes>
            <Route path="/" element={<Login/>}></Route>
            <Route path="Submit" element={<Submit/>}></Route>
         </Routes>
         <Footer/>
      </div>
   )
}

export default App ;
```

Output screen for App.js

Fig 7.1.4 output for App.js

7.1.5 Login.js

```
import React from "react"
import {useNavigate} from "react-router-dom";
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import axios from 'axios';

function Login(){
   const navigate = useNavigate();
```

//Use state is used when the variables get constantly updated.Initially we can specify an empty value or any default one.

```
   const [value,setChange]=React.useState({
      email : "",
      location : "",
      job : "",
      time : "00:00",
      day : "sunday"
   });
```

//When the user enters the input, this function gets triggered and updates it wil the current value.

```
   function changeValue(event){
      const{name,value}=event.target
      setChange((prevValue)=>{
```

```
            return{
                ...prevValue,
                [name]:value
            }
        })


    }
    console.log(value);
```

//When the button is clicked this function gets triggered, and if no errors it navigates to the submit page

```
async function handleClick()
{

    if (value.email==="" || value.location ===""  || value.job==="" || value.time===""){
            toast.error("Fill all the details", {
                theme:"colored",
                autoClose:1000,
                position: "top-center",
                draggable:true,
                pauseOnHover:true,
            }
        )
    }
    else{
            navigate("Submit")
    }
```

//Sending all the frontend inputs to the backened with the help of axios

```
    try{
        await axios.post("http://localhost:4000/post_name",{
            e : value.email,
            l : value.location,
            j : value.job,
            t : value.time,
            d : value.day
        })
    } catch(error){
        console.log(error)
    }

}
    return(
```

// All the inputs given to the user are written below

```
<div className="login">
  <form>
    <input
      placeholder="Email address"
      name = "email"
      type= "email"
      value={value.email}
      onChange={changeValue}
      />
    <input
      placeholder="Location"
      name = "location"
      value={value.location}
      onChange={changeValue}
      />
    <input
      placeholder="Job role"
      name = "job"
      value={value.job}
      onChange={changeValue}
      />
    <div className="automation">
      <label>Scheduling cycle</label>
      <br/>
      <select className="time"
        name = "day"
        value={value.day}
        onChange={changeValue}>
        <option id="0" value="sunday">Sunday</option>
        <option id="1" value="monday">Monday</option>
        <option id="2" value="tuesday">Tuesday</option>
        <option id="3" value="wednesday">Wednesday</option>
        <option id="4" value="thursday">Thursday</option>
        <option id="5" value="friday">Friday</option>
        <option id="6" value="saturday">Saturday</option>
      </select>
      <br/>
      <input className="time"
        type="time"
        name = "time"
        value={value.time}
        onChange={changeValue}
        />
    </div>
  </form>
```

```
            <button onClick={handleClick}>Submit</button>
            <ToastContainer />
        </div>
    )
}
export default Login;
```

Output for Login.js

Fig 7.1.5 output for Login.js

7.1.6 Header.js

```
import React from "react";

//The header component of the page

function Header(){
    return(
        <header>
            <h1>Online Job Notifier</h1>
        </header>
    );
}

export default Header;
```

Output for Header.js

Fig 7.1.6 output for Header.js

7.1.7 Footer.js

```
import React from "react";

//The footer component of the page

function Footer(){
   const year = new Date().getFullYear();
   return(
      <footer>
         <p>Copyright {year}</p>
      </footer>
   );
}

export default Footer;
```

Output for Footer.js

Fig 7.1.7 Footer.js

7.1.8 Submit.js

```
import React from "react";

//This page gets loaded as soon as the user clicks on submit button

function Submit(props){

   return(
      <div className = "submitPage">
         <h1>Thankyou!</h1>
         <p>Your Information is recorded and you'll be receiving updates as per the
automation time you entered.</p>
      </div>
   );
}

export default Submit;
```

Output for submit.js

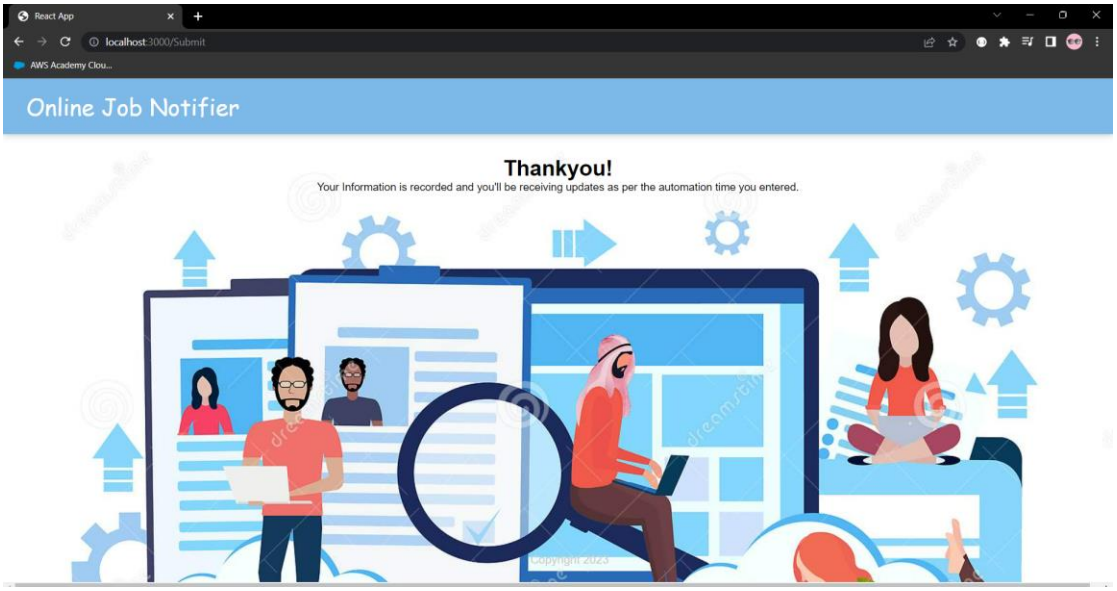


Fig 7.1.8 output for Submit.js

7.1.9 styles.css

```
* {
  padding: 0;
  margin: 0;
  box-sizing: border-box;
}
html {
  font-family: "Montserrat", sans-serif;
}
body {
  background: #eee;
  padding: 0 16px;
  background-image: url(https://thumbs.dreamstime.com/z/mix-race-people-searching-job-concept-human-resource-manager-choose-resume-cv-professional-candidate-flat-male-cartoon-mix-race-126959610.jpg);
  background-repeat: no-repeat;
  background-size:cover;
}

header {
  background-color: #7CB9E8;
  margin: auto -16px;
  padding: 16px 32px;
  box-shadow: 0 0 10px 0 rgba(0, 0, 0, 0.3);
}

header h1 {
  color: #fff;
```

```
  font-family: "McLaren", cursive;
  font-weight: 200;
}

.login{
  text-align: center;
  padding:20px;
}

input{
  margin-left:50px;
  margin-right:50px;
  width:370px;
  padding:5px;
}
.automation{
  margin-top: 20px;
}

button{
  padding:5px;
  margin-top:15px;
  background-color: #AFDBF5;
  width:200px;
}

.submitPage{
  text-align:center;
  /* color:gray; */
  font-size:15px;
  padding:30px;
}
footer {
  position: absolute;
  text-align: center;
  bottom: 0;
  width: 100%;
  height: 2.5rem;
}

footer p {
  color: #ccc;
}
.note {
  background: #fff;
  border-radius: 7px;
  box-shadow: 0 2px 5px #ccc;
```

```
  padding: 10px;
  width: 240px;
  margin: 16px;
  float: left;
}
.note h1 {
  font-size: 1.1em;
  margin-bottom: 6px;
}
.note p {
  font-size: 1.1em;
  margin-bottom: 10px;
  white-space: pre-wrap;
  word-wrap: break-word;
}

label{
  color:gray;
  font-size:15px;
}
.time{
  margin-top:5px;
}
```

7.2 EXECUTION PROCESS:

STEP 1: Open my-app folder where all the react files are stored, we can use command prompt or hyper terminal to run the react files. Upon opening the hyper terminal then enter the command **npm start.**

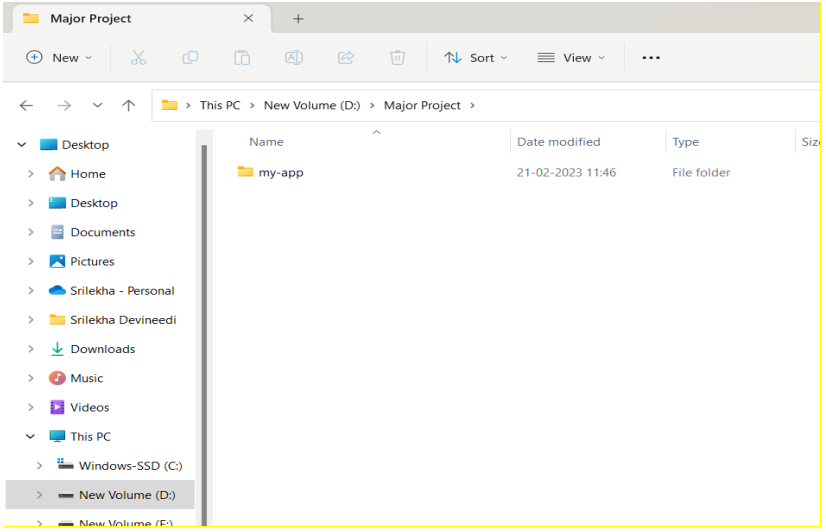


Fig 7.2 my-app folder path

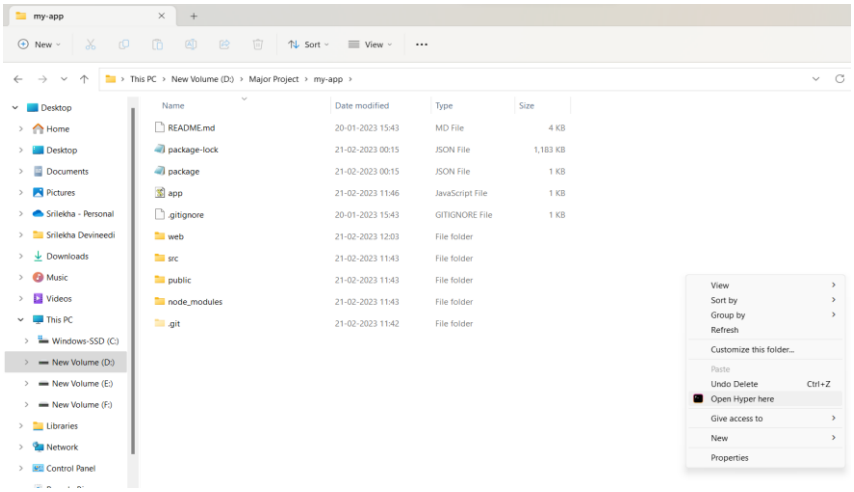


Fig 7.2 files in my-app folder

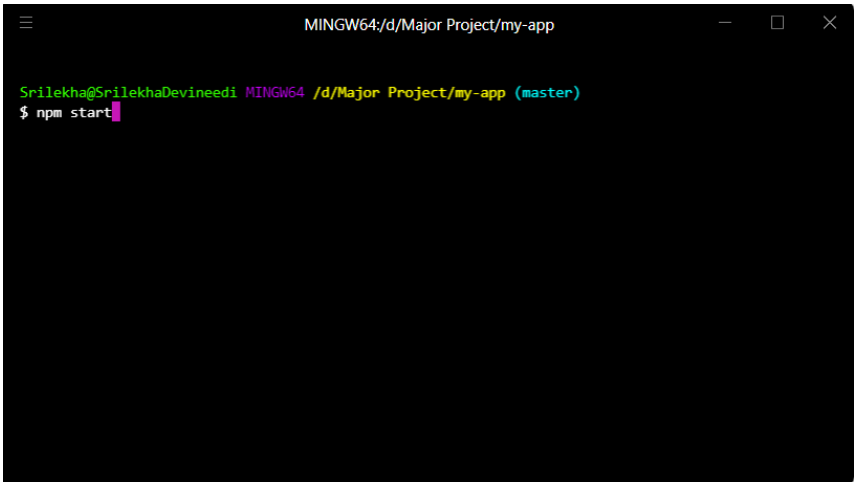


Fig 7.2 hyper terminal to start the execution

The command **npm start** refers to process the react files and to open the website

STEP 2: After the processing and initializing of react files the website opens with successful compilation, if the website is unable to open then we can open the website through the link provided in the hyper terminal when compiled.

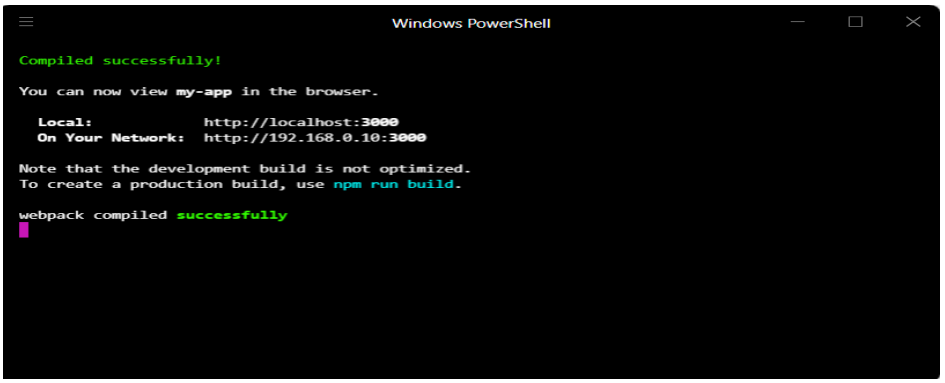


Fig 7.2 compiling webpage

Fig 7.2 frontend code webpage

The above website is the interface where the user provides the inputs, now to retrieve the information that user gave, we need to start the backend server.

STEP 3: To run the backend code we need to open another hyper terminal at the location where the backend code resides.

my app -> web -> app.js

STEP 4: Enter the command **nodemon app.js** in the terminal, we use nodemon because the server gets restarted as soon as we make a change in the code and there is no need for the developer to restart it all the time.

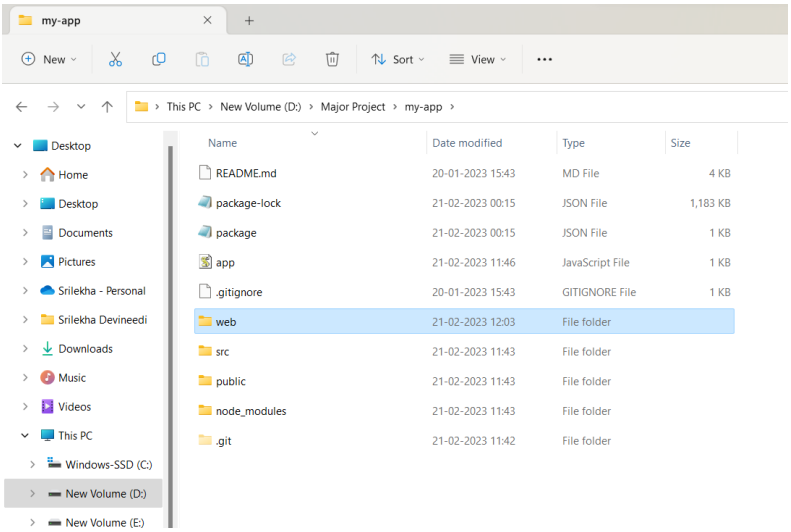


Fig 7.2 web folder in my-app

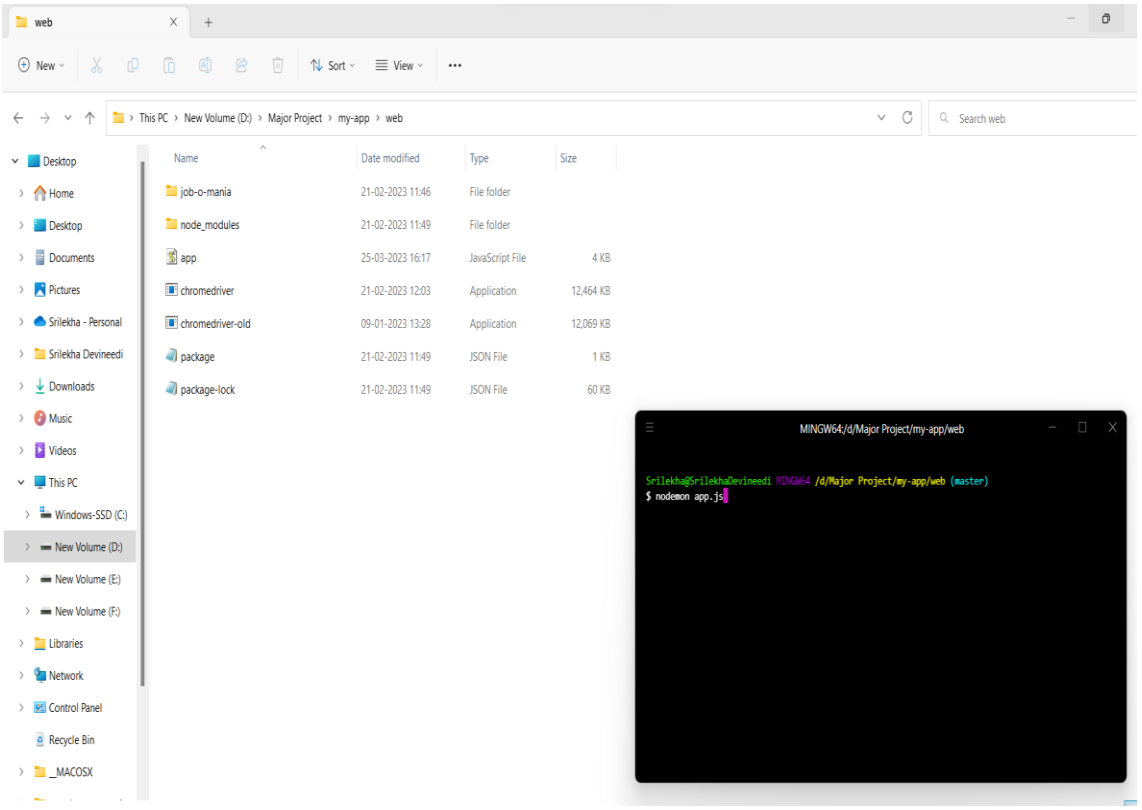


Fig 7.2 hyper terminal in web folder

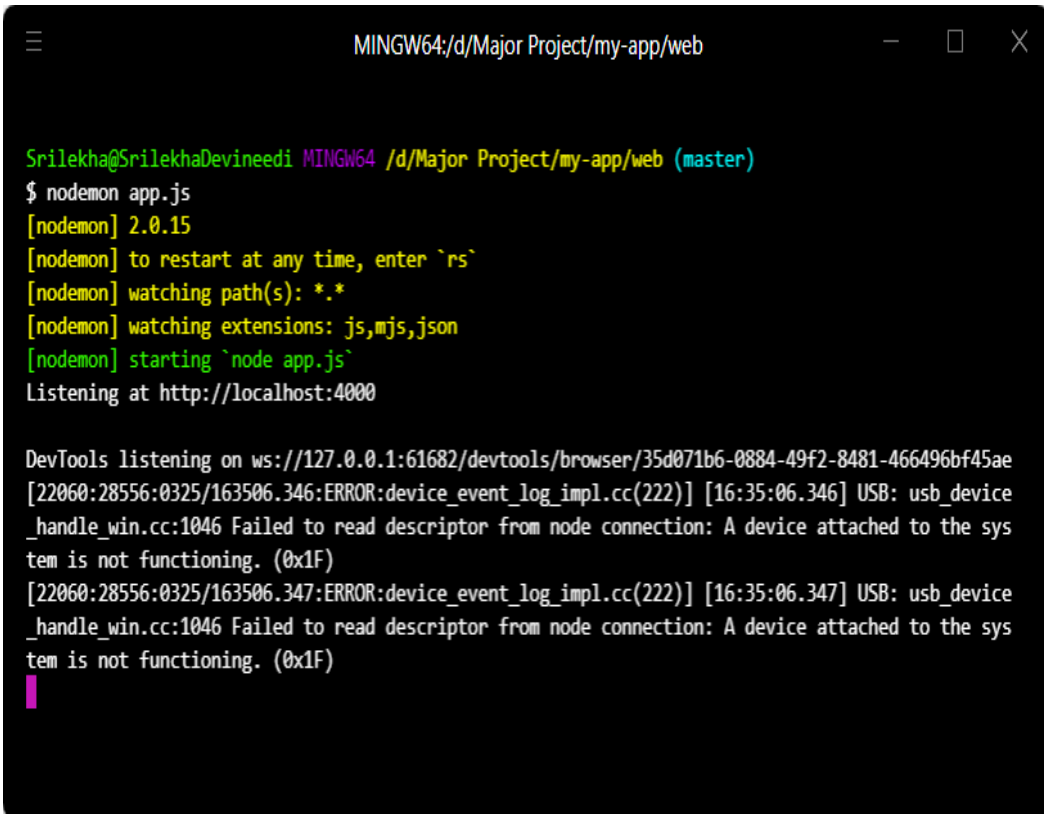


Fig 7.2 server getting started

STEP 5: After the backend server gets started, the user gives the inputs through the website.

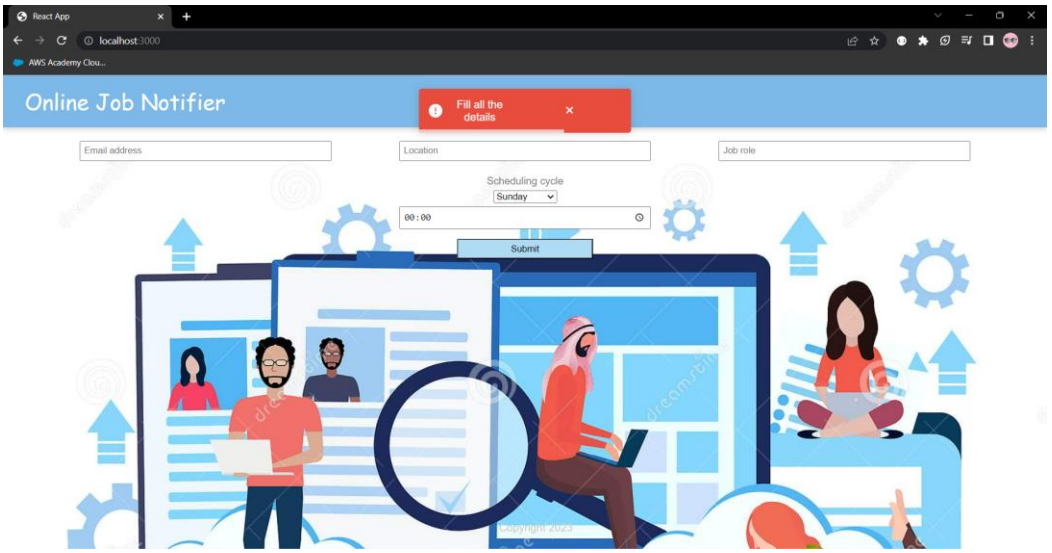


Fig 7.2 backend webpage

STEP 6: User should enter the details by giving their email, preferred location, and preferred job role according to their skills. If user left any input empty, then there will be a alert displayed stating “fill all the details”.

User can schedule the time cycle to get the notification regarding the job vacancies. Here cron-job scheduler is used to schedule tasks to run on the server to complete repetitive tasks automatically.Where the cron-job takes 6 inputs like seconds, minutes, hours, day of month, month, day of week.

In our project we have mentioned week wise scheduling, and also preferred time to the user to get notifications.

Fig 7.2 backend webpage

STEP 7: After filling the details it renders to submit page that the details of the user are recorded and will receive updates as per the scheduling time the user has entered .

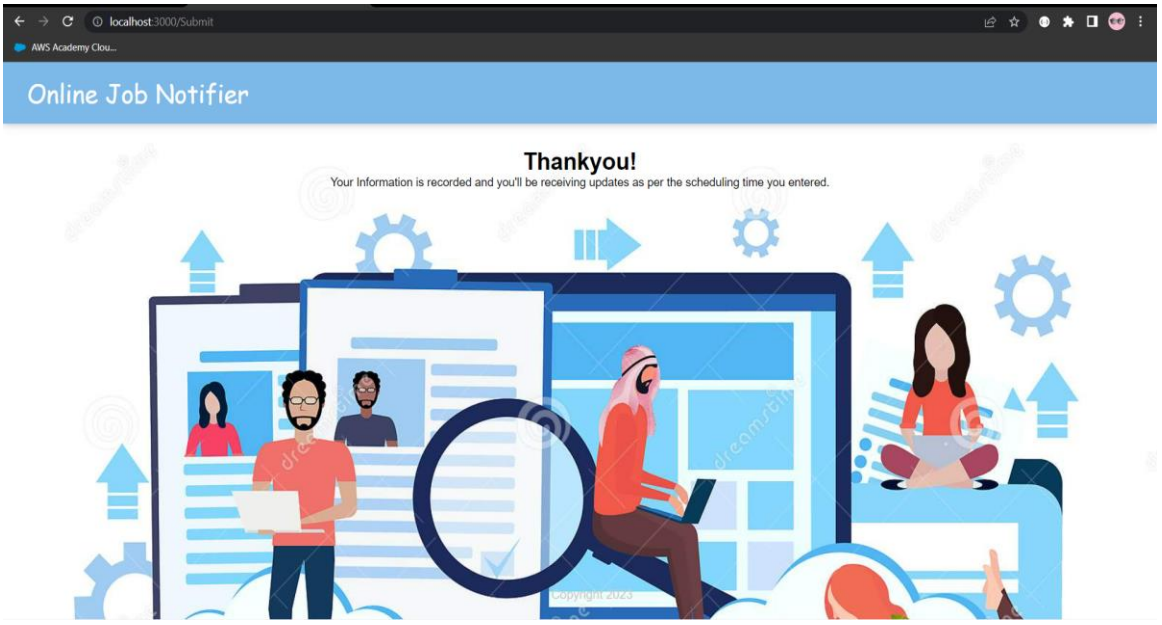


Fig 7.2 submit page

STEP 8: Now we can check in the backend terminal whether the mail is sent or not to the user. The inputs given by the user will be displayed in the terminal and also the information regarding the job vacancies will also be displayed in the terminal and at last the user mail to which the notification has to be sent will be shown and if the notification is sent then in the terminal, it updates that it is sent to the user as shown below.

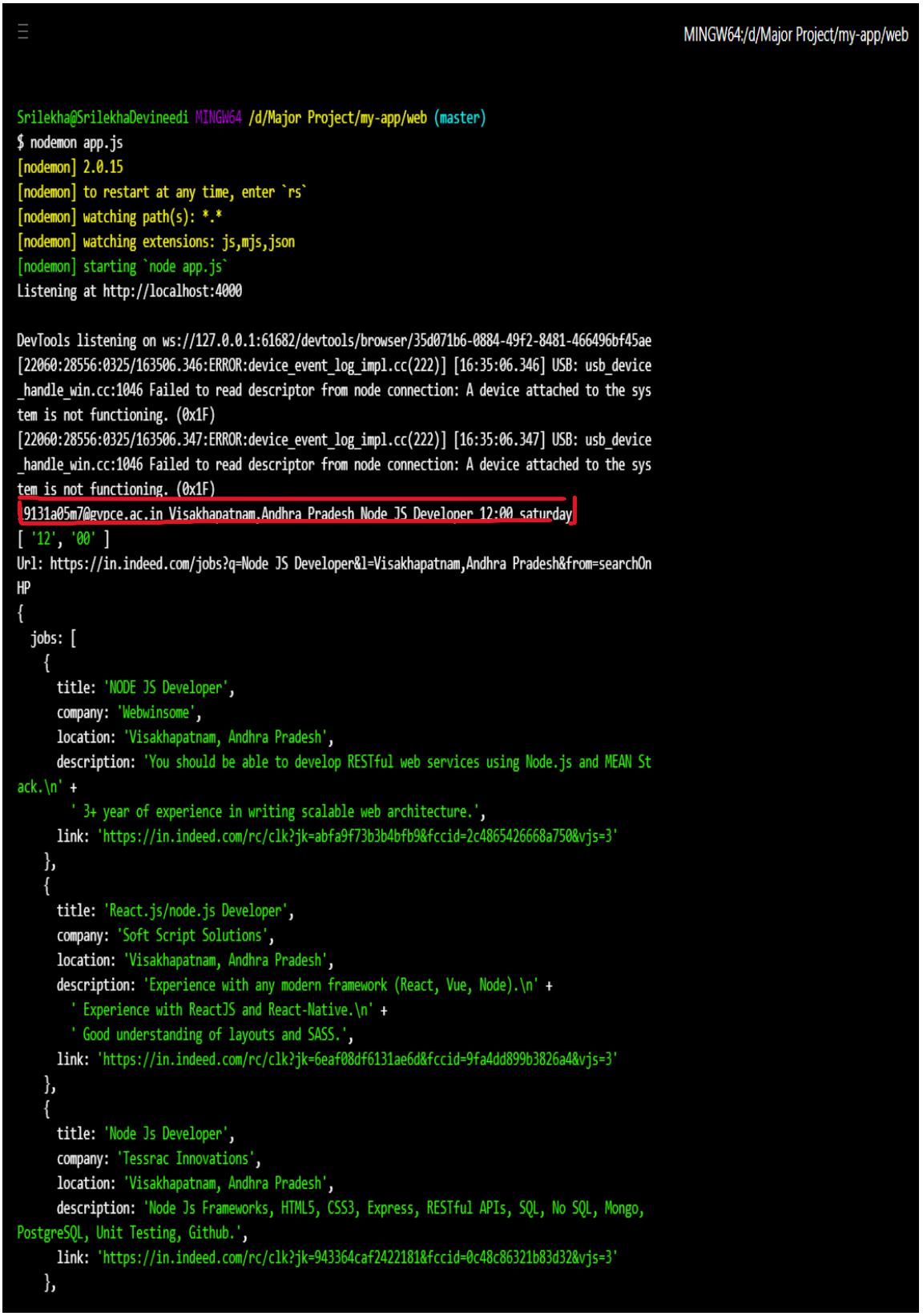


Fig 7.2 details of job-openings

Fig 7.2 mail sent info in terminal

7.3 RESULTS:

INPUT OUTPUT SCREENS:

EMAIL NOTIFICATION

The user will receive the notification regarding job vacancies.

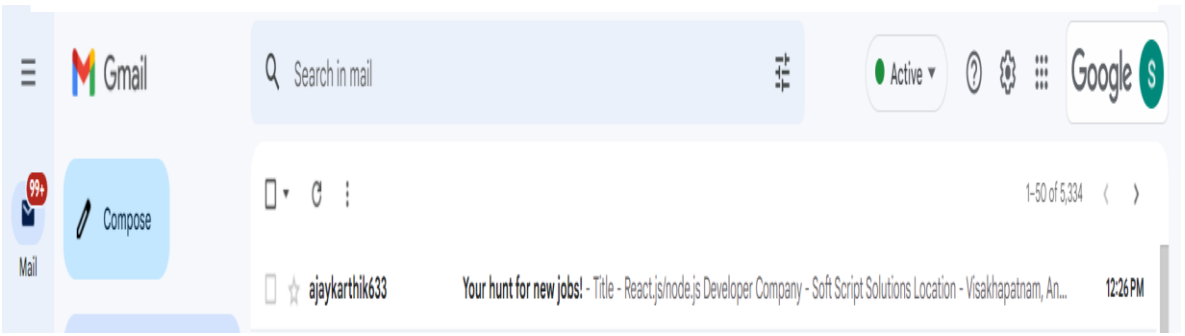


Fig 7.3 mail notification

INFORMATION REGRADING NEW JOB OPENINGS:

The information regarding the job openings will be notified with different job openings with the links and also the location mentioned by the user and also the company details will also be given in the notification.

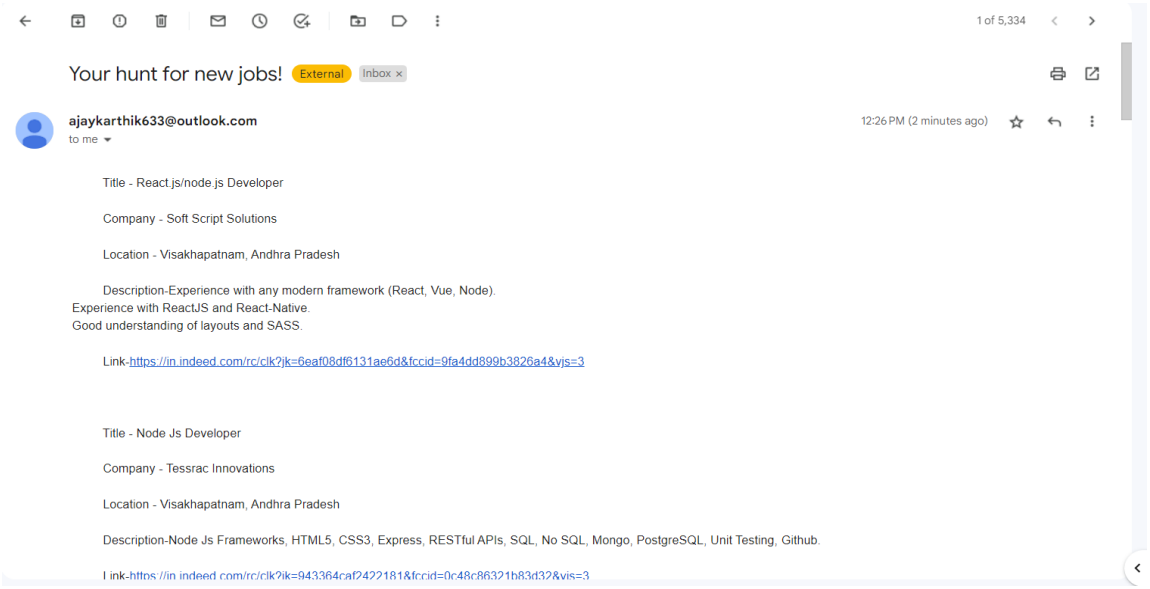


Fig 7.3 information in mail

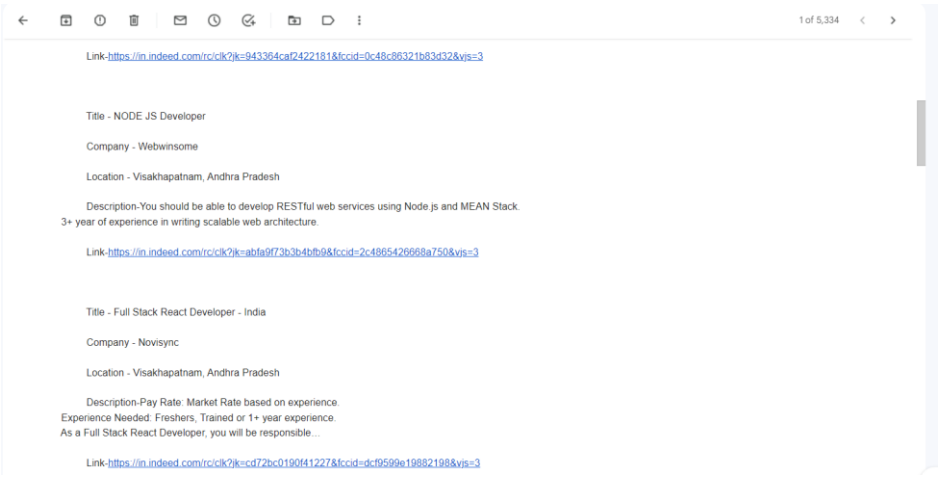


Fig 7.3 information in mail

8. TESTING

8.1 INTRODUCTION

SOFTWARE TESTING is defined as an activity to check whether the actual results match the expected results and to ensure that the software system is Defect free. It involves the execution of a software component or system component to evaluate one or more properties of interest. It is required for evaluating the system. This phase is the critical phase of software quality assurance and presents the ultimate view of coding.[10]

Importance of testing:

The importance of software testing is imperative. A lot of times this process is skipped, therefore, the product and business might suffer. To understand the importance of testing, here are some key points to explain.

- ➢ Software Testing saves money
- ➢ Provides Security
- ➢ Improves Product Quality
- ➢ Customer satisfaction

Testing is of different ways The main idea behind the testing is to reduce the errors and do it with a minimum time and effort.

Benefits of Testing:

Cost-Effective: It is one of the important advantages of software testing. Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix.

Security: It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.

Product quality: It is an essential requirement of any software product. Testing ensures a quality product is delivered to customers.

Customer Satisfaction: The main aim of any product is to give satisfaction to their customers. UI/UX Testing ensures the best user experience.

Testing can be done in different ways. Some of the types of testing are mentioned below. The main purpose of any type of test is to systematically uncover different classes of errors and do so with a minimum amount of time and effort.

Types of testing:

- Unit testing
- Integration testing

- Regression testing
- System testing
- Performance Testing etc..

Testing can be done manually or by testing tools. There are several testing tools for different software.

Unit Testing: It is a method by which individual units of source code, sets of one or more computer modules together with associated control data, usage procedures and operating procedures, are tested to determine if they are fit for use. Unit testing focuses verification 31 effort on the smallest unit of software design-the software component or module.

Integration Testing: It is the phase in software testing in which individual software modules are combined and tested as a group. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

Regression Testing: Regression testing is any type of software testing that seeks to uncover new software bugs, or regressions, in existing functional and non-functional areas of a system after changes such as enhancements, patches or configuration changes, have been made to them.

System Testing: System testing of software or hardware is testing conducted on a complete integrated system to evaluate the system's compliance with its specified requirements. System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system.

Performance Testing: Performance testing is designed to test the run-time performance of software within the context of an integrated system. Performance testing occurs throughout all steps in the testing process. Even at the unit level, the performance of an individual module may be assessed as white-box tests are conducted. This project reduces manual work and helps in getting students attendance status much faster.

8.2 TEST CASES

First of all let's test the code with some random inputs by specifying the job role as Node JS Developer and location as Visakhapatnam, Andhra Pradesh. After few seconds the mail will be sent to the id mentioned by the user. When the user opens the email, there will be enumerous number of jobs listed along with the links of indeed website. User can go with their own choice, go through the link and apply to the company. As soon as the user opens the links there are many details mentioned like the salary, job type, qualifications, benefits, prerequisites, about their company etc.

INPUTS GIVEN BY THE USER:

8.2.1 TEST CASE 1:

This is the frontend code output where the user should give the inputs.

Fig 8.2.1 frontend code webpage

OUTPUT:

information regarding job openings

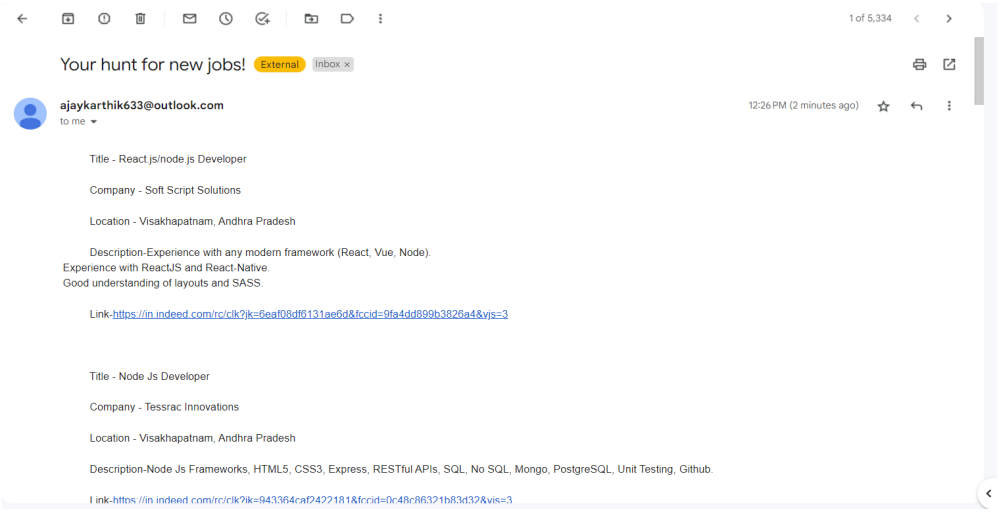


Fig 8.2.1 information in mail

On opening the link the above are the outputs for two job openings as tested for the project.

JOB OPENING 1:

The jobseeker can apply for the job opening with required skills and all the details regarding the job will be described.

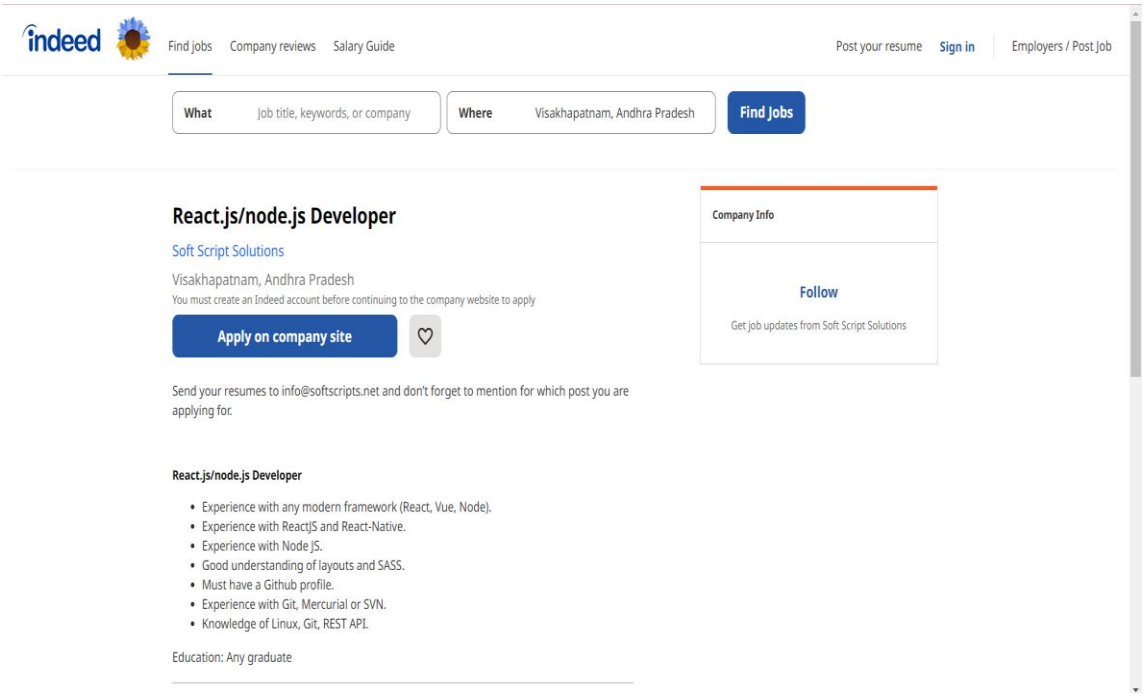


Fig 8.2.1 job-opening 1

JOB OPENING 2:

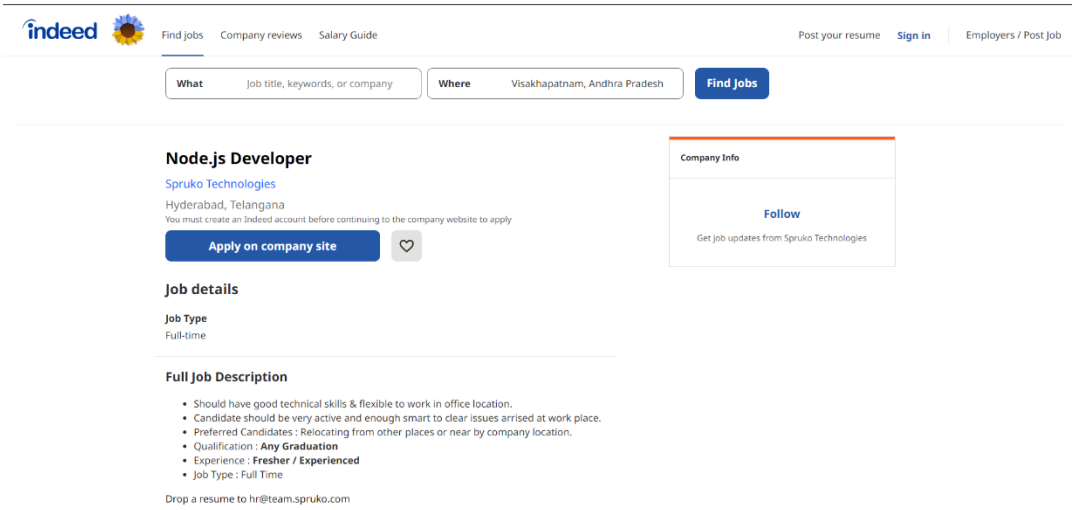


Fig 8.2.1 job-opening 2

The second test case includes other job title, with another location. Again the procedure remains unchanged and the user can check the mail and select their desired role and the scheduling cycle is completely dependent on users wish. The user can set the scheduling time on the day and time when they are free. So as per the time they have given, they will be receiving updates every week.

Same like this, we can mention any location specifying the state as well and we can retrieve jobs.

INPUTS GIVEN BY THE USER:

8.2.2 TEST CASE 2:

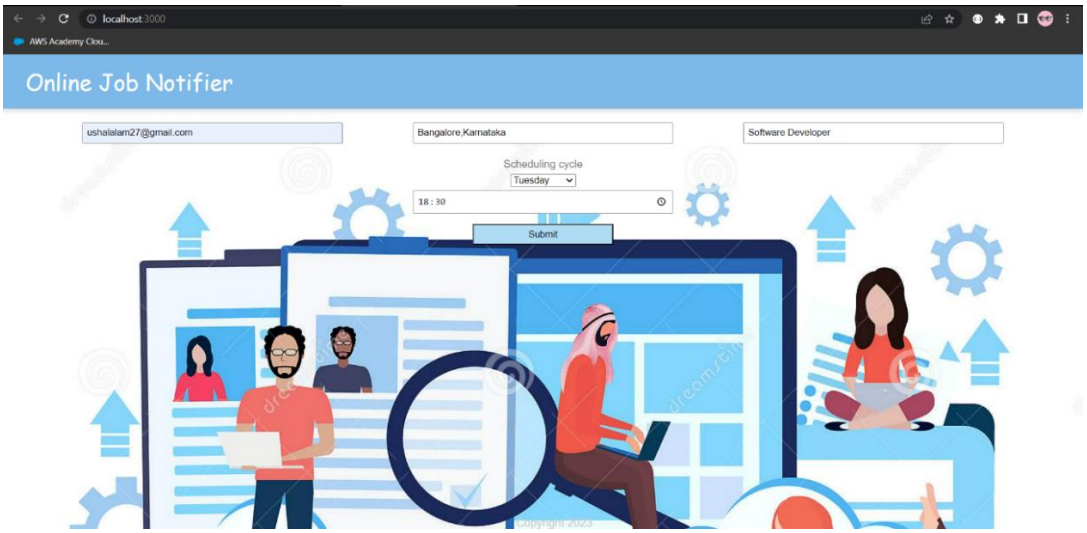
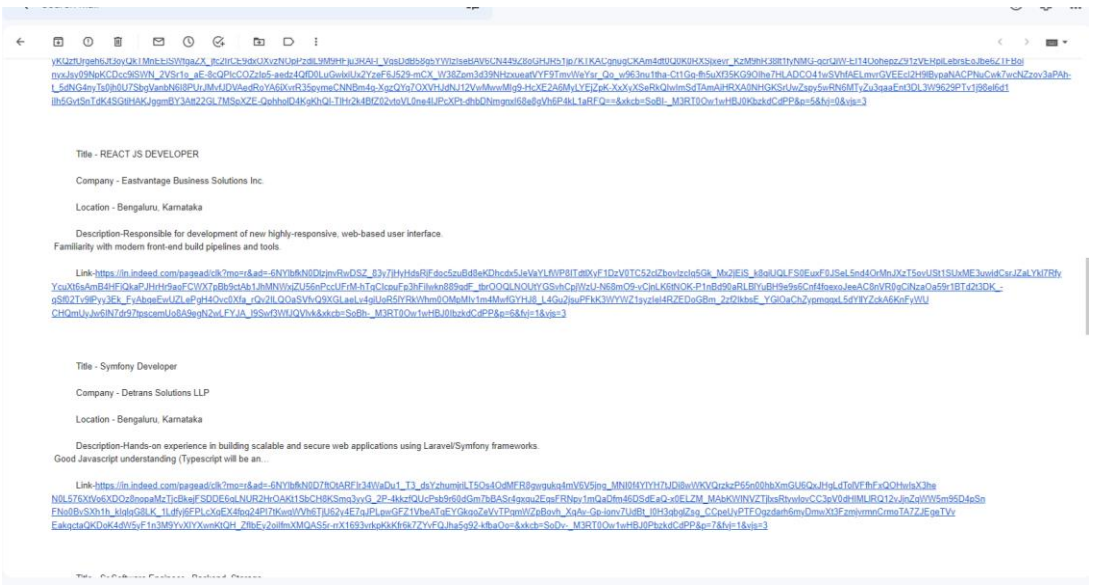


Fig 8.2.2 frontend webpage

OUTPUT:

information regarding job openings



Title - REACT JS DEVELOPER

Company - Eastvantage Business Solutions Inc.

Location - Bengaluru, Karnataka

Description-Responsible for development of new highly-responsive, web-based user interface.
Familiarity with modern front-end build pipelines and tools.

Title - Symfony Developer

Company - Detrans Solutions LLP

Location - Bengaluru, Karnataka

Description-Hands-on experience in building scalable and secure web applications using Laravel/Symfony frameworks.
Good Javascript understanding (Typescript will be an...

Fig 8.2.2 information in mail

On opening the link the above are the outputs for two job openings as tested for the project.

JOB OPENING 1:

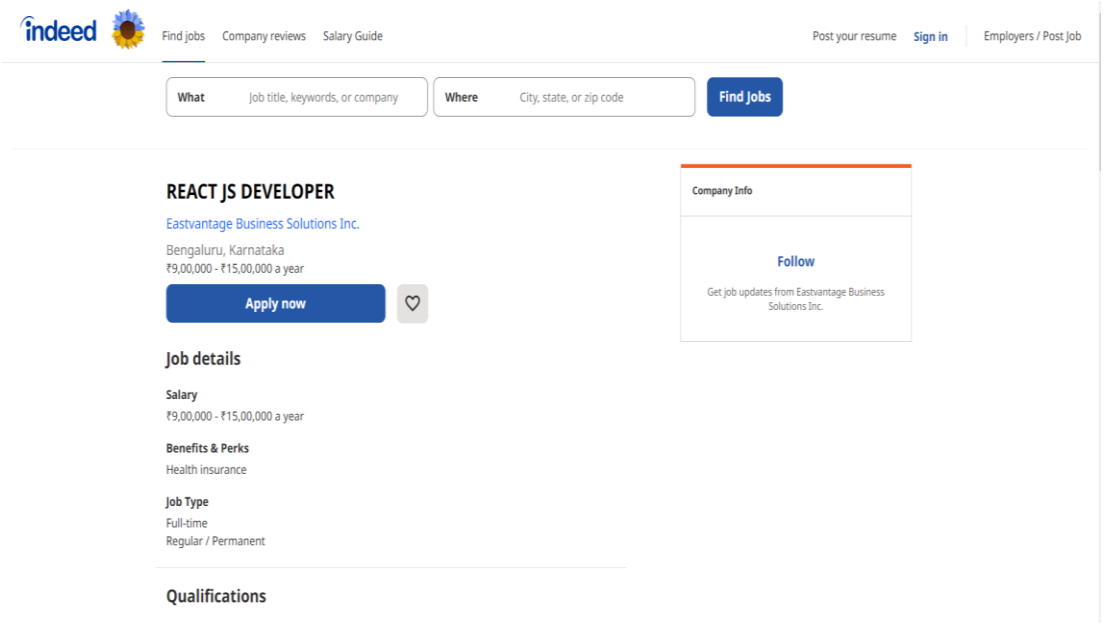


Fig 8.2.2 job-opening 1

JOB OPENING 2:

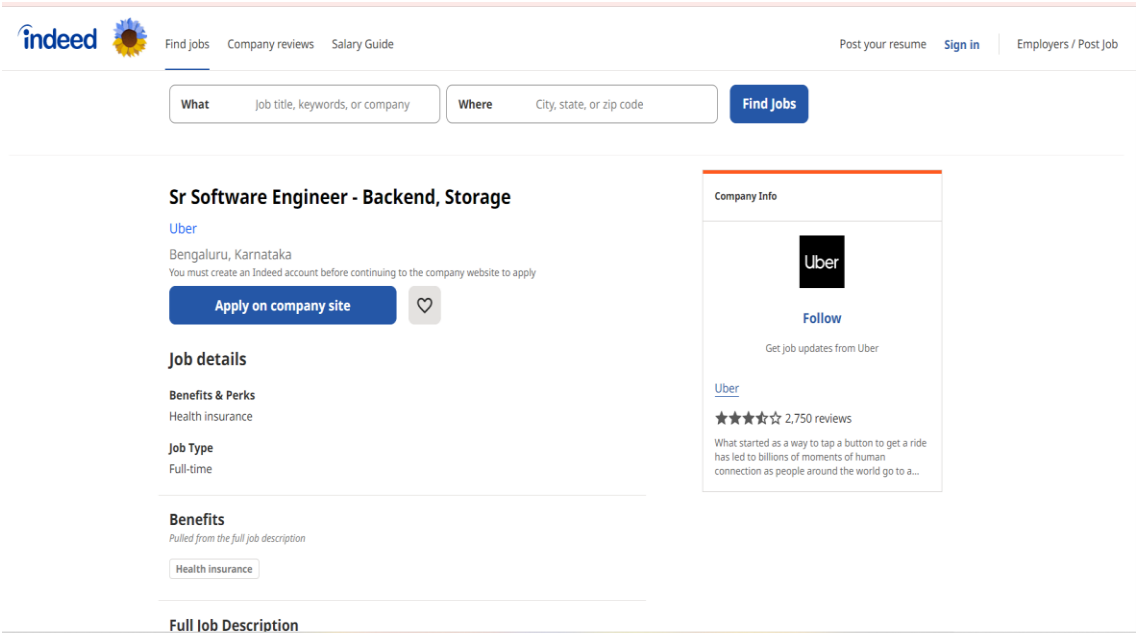


Fig 8.2.2 job-opening 2

8.2.3 TEST CASE 3:

The indeed website works only for searching jobs in india, so here we've given another country as location input and we can see that the output job arrays are empty and also the email is empty, because there is no data that is matching to our input.

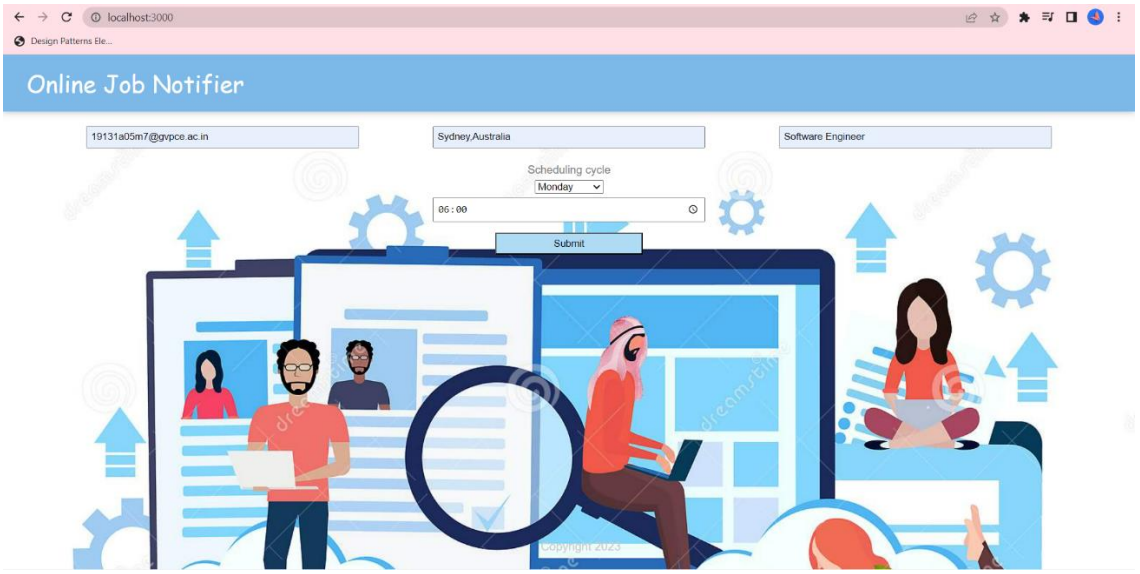


Fig 8.2.3 frontend code webpage

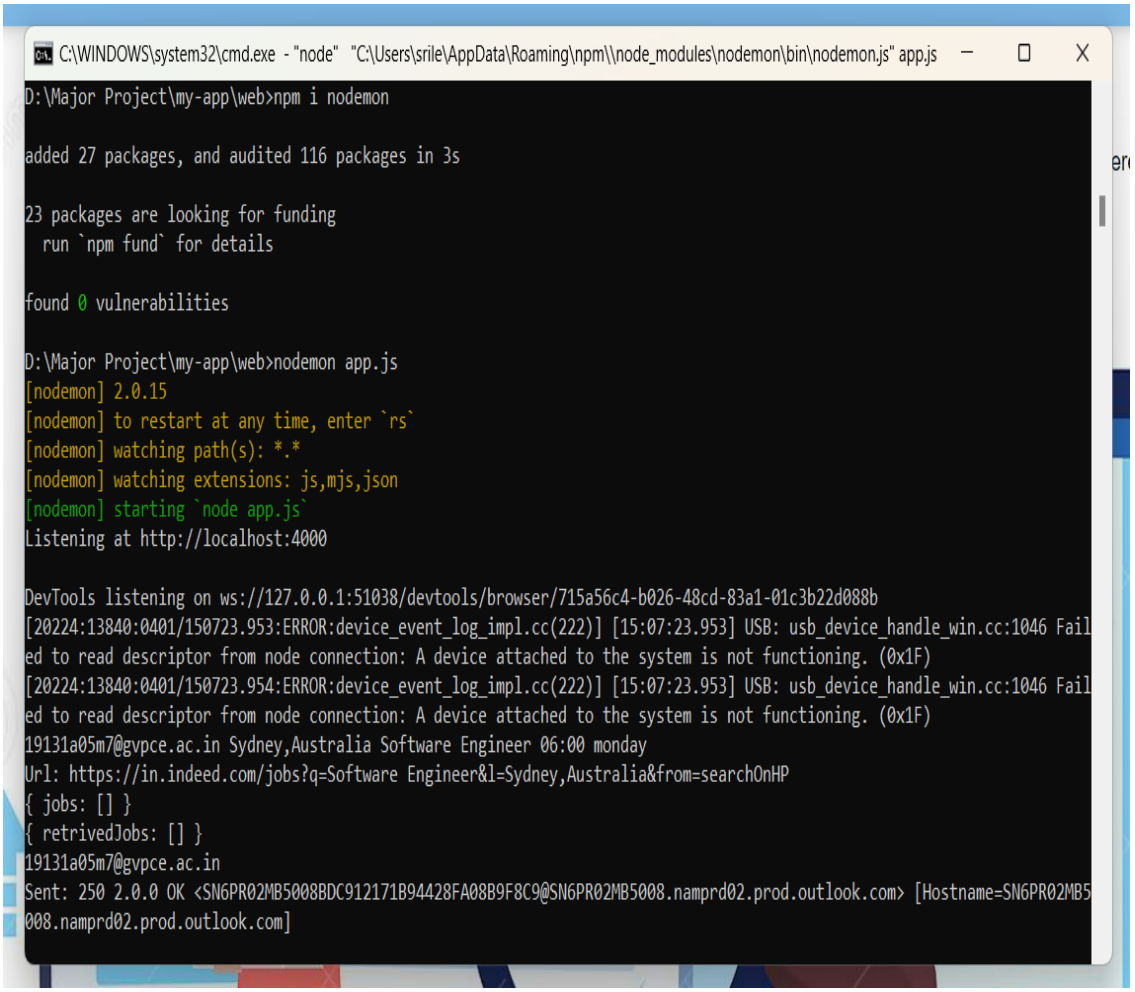


Fig 8.2.3 hyper terminal

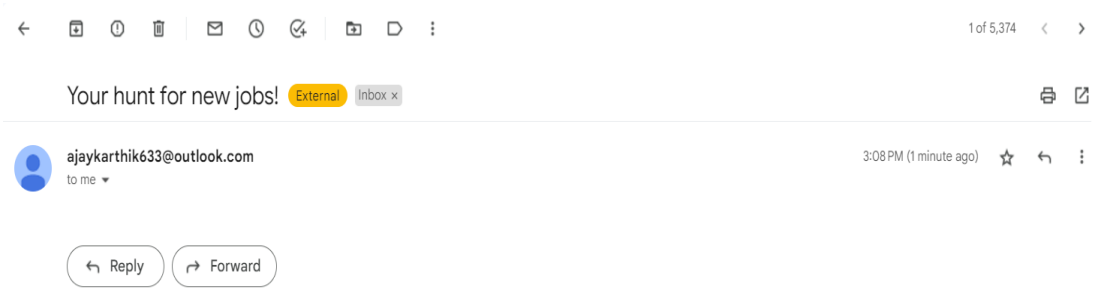


Fig 8.2.3 mail notification

9. CONCLUSION

The wider areas of job searching facilitate the quick and easy access to opportunities. The increasing job opportunities and changing scenario of the business environment today has made more people to search for better career and employers to search for better potential. This situation has prompted many to move to job portals to look for the ways that has been widely accepted and fully useful in job searching. In this sense the job portals assume greater importance and we could develop such an efficient system which is used by lot many job hunters and employers.

This project is developed using web scraping model which will rescue at that point where user should spend a lot of time. The entire process of web scraping as well as job searching manually becomes faster when it is replaced by automated processes. Candidates are just a few clicks away to get connected. This application will be helpful in finding new openings on job portals and alert the user by providing required information.

Every field of user responses are taken into consideration and web scraping in the indeed portal is done accordingly. The updates regarding the jobs are automated every week so that there is no need for the user to search for them manually.

10. FUTURE SCOPE

There is ample scope of enhancement and adding functionalities to this application. This application can be extended to send automated interview scheduling acceptance/rejection of resume. The application can have a job recommendation system based on frequent search result of different users. There can be a feedback or review section for the application. The application can be more scalable by extending the search functionality based on country, city or area. The User Interface can be made more attractive and user friendly.

11. REFERENCES

[1] Vivek Kumar Sehgal1, “Job Portal-A Web Application for Geographically Distributed Multiple Clients”, 1 Department of CSE and ICT Jaypee University of Information Technology, Waknaghat, Solan, H.P (INDIA)M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

[2]https://seanholbert.wor dpress.com/2011/07/15/scrappy-simple-web scraping/, Jun.2015”

[3] https://www.datahen.om/3advantages-web-scraping enterprise/,May.17,2017”

[4] https://www.seobility.net/en/wiki/CronJob

[5]https://ieeexplore.ieee.org/search/searchresult.jsp?newsearch=true&queryText=web%20scrapping

[6] https://www.npmjs.com/package/selenium-webdriver

[7] https://www.npmjs.com/package/cheerio

[8] https://en.wikipedia.org/wiki/Systems_architecture

[9] https://www.tutorialspoint.com/uml/uml_statechart_diagram.htm

[10] https://www.guru99.com/software-testing-introduction-importance.html