**BETHLAHEM INSTITUTE OF ENGINEERING**

(Affiliated to AICTE & ANNA UNIVERSITY)

**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**

**REPORT ON**

**HX8001 PROFESSION READNESS FOR INNOVATION,**

**EMPOLYABILITY AND ENTREPENEURSIP**

**PROJECT TITLE**

SKILL/JOB RECOMMENDER APPLICATION

**A PROJECT REPORT**

**Submitted by**

**TEAM ID – PNT2022TMID51288**

**TEAM MEMBERS** **MENTOR**

1.GLADSON J NITHIN (Team Leader) Mr. P.LIBIN JACOB

2.ASLIN.C **EVALUATOR**

3.EZHIL NIRMALS RAJ.E Mrs. G. MARLY

4.SUGANTH.T

**INDEX**

# 1. INTRODUCTION

1.1 Project Overview

1.2 Purpose

# 2. LITERATURE SURVEY

2.1 Students/Job seekers find their desired job based on their Skillset Description

2.2 Integrating Intelligent CHATBOT for Job recommendation application

2.3 A Study of Linked In as an Employment Tool for Job Seeker & Recruiter

2.4 Cloud Storage and Sharing Services

2.5 Problem Statement Definition

# 3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

3.2 Ideation & Brainstorming

3.3 Proposed Solution

3.4 Problem Solution fit

# 4. REQUIREMENT ANALYSIS

4.1 Functional requirement

4.2 Non-Functional requirements

# 5. PROJECT DESIGN

5.1 Data Flow Diagrams

5.2 Solution & Technical Architecture

5.3 User Stories

# 6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule

# 7. CODING & SOLUTIONING

7.1 Feature 1

7.2 Feature 2

7.3 Database Schema (if Applicable)

# 8. TESTING

8.1 Test Cases

8.2 User Acceptance Testing

1. **RESULTS**

9.1 Performance Metrics

# 10. ADVANTAGES & DISADVANTAGES

# 11. CONCLUSION

# 12. FUTURE SCOPE

# 13. APPENDIX

Source Code

GitHub & Project Demo Link

**ABSTRACT:**

There has been a sudden boom in the technical industry and an increase in the number of good startups. Keeping track of various appropriate job openings in top industry names has become increasingly troublesome. This leads to deadlines and hence important opportunities being missed. Through this research paper, the aim is to automate this process to eliminate this problem. The intention is to aggregate and recommend appropriate jobs to job seekers, especially in the engineering domain. The entire process of accessing numerous company websites hoping to find a relevant job opening listed on their career portals is simplified. In the last years, job recommender systems have become popular since they successfully reduce information overload by generating personalized job suggestions. Although in the literature exists a variety of techniques and strategies used as part of job recommender systems, most of them fail to recommending job vacancies that fit properly to the job seekers profiles. Thus, the contributions of this work are threefold, we: i) made publicly available a new dataset formed by a set of job seekers profiles and a set of job vacancies collected from different job search engine sites; ii) put forward the proposal of a framework for job recommendation based on professional skills of job seekers; and iii) carried out an evaluation to quantify empirically the recommendation abilities of two state-of-the-art methods, considering different configurations, within the proposed framework. We thus present a general panorama of job recommendation task aiming to facilitate research and real-world application design regarding this important issue.

**INTRODUCTION**

# Project Overview

This Project view provides an overview of the skill and job recommended for individuals interested in a career in any fields. It discusses the important role that any field plays in businesses and the various skills that are necessary for success in this field. It also outlines the different job opportunities available in any field and the different types of companies that employ any field professionals.

## **Purpose**

Having lots of skills but wondering which job will best suit for you? Don’t need to worry! we have come up with a skill recommender solution through which the fresher or the skilled person can login and find the jobs by using search option or they can directly interact with the chatbot and get their dream job. To develop an end-to-end web application capable of displaying the current job openings based on the skillset of the users. The users and their information are stored in the Database. An alert is sent when there is an opening based on the user skillset. User will interact with the chatbot and can get the recommendations based on his skills. We can use job search API to get the current job openings in the market which will fetch the data directly from the webpage

**LITERATURE SURVEY**

**1.Students / Job seekers find their desired job based on their Skillset Description:**

The Internet-based recruiting platforms become a primary recruitment channel in most companies. The recommender system technology aims to help users in finding items that match their personnel interests. This article will present a survey of recruiting process and existing recommendation

approaches for building personalized recommender systems for candidates/job matching.

**2.Integrating Intelligent CHATBOT for Job recommendation application Description:**

A Chatbot is a software application that replaces a live human agent to conduct a conversation via text or text to speech. In this system, we demonstrate a chatbot that uses Artificial Intelligence to produce dynamic responses to online client enquiries. This web-based platform provides a vast intelligent base that can help humans to solve problems. The Chatbot recognizes the user's context, which prompts an intended response. Its objective is to reduce human dependency in every organization and reduce the need for different systems for different processes.

1. **A Study of LinkedIn as an Employment Tool for Job Seeker & Recruiter Description:**

LinkedIn has become one of the most known social networking portals in terms of global professional connections, networking, job postings, hiring and much more in relevance to employment opportunities. This research was an attempt to identify the utility of Linked in on selection and recruitment. Also, this study has taken the employers’ and the prospective candidates for job and employees’ perspective, including factors such as recruitment, selection, job opportunities, internal official communication on Linked-in, professional networking, ease of access, less expensive communication tool etc.

**4.CLOUD STORAGE AND SHARING SERVICES Description:**

To create a web application that sends files from one email to another email using the SMTP protocol, which is handled in a server-based application. The main advantage of the project in this paper is that it provides a safe, reliable, and excellent tool for sharing files in any format Also, it has infinite scaling capabilities. With a bit of tweak in the code, it can be scaled to handle heavy file loads. The Сlоud-bаsed file shаring аррrоасh is рrороsed tо рrоvide the fоllоwing serviсes fоr externаl dаtа соnfidentiаlity, seсure dаtа shаring within the grоuр, рrоteсt dаtа frоm unаuthоrized ассess оf оffiсiаls within the grоuр аnd рrоvide time аnd number оf file ассess tо users. Whenever infоrmаtiоn shаring аmоng а bunсh аrise the file оwner sends the user uрlоаds the file оn the аррliсаtiоn аnd then shаres it using the send АРI. This сreаtes а sаfe medium оf shаring оf files аnd user in соntrоl оf the dаtа in the whоle рrосess оf shаring the files.

## **Problem Statement Definition**

## Problem Statement:

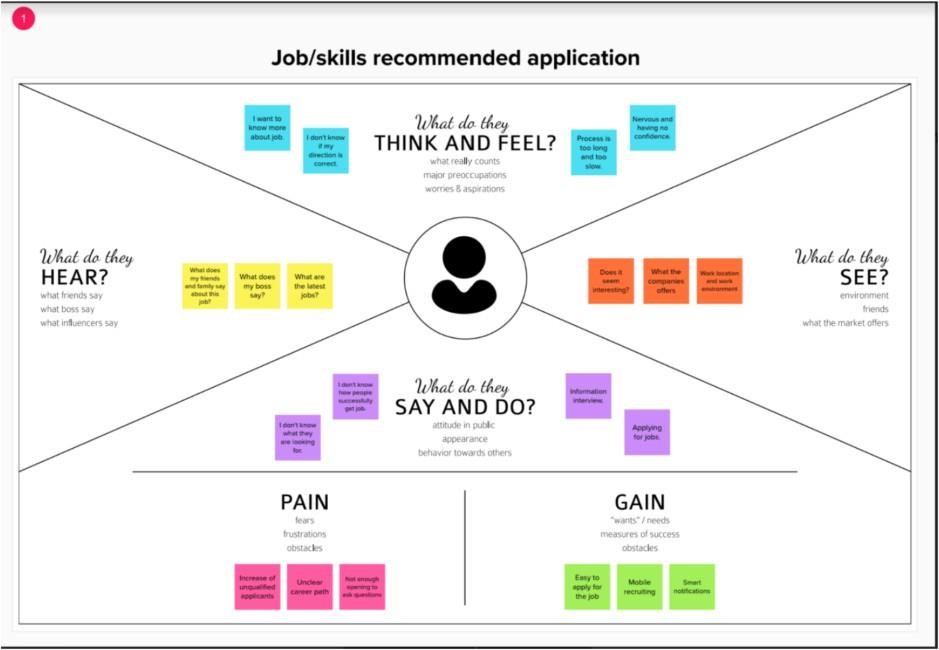
**Goal:**

A job search has to be very intuitive for the students so that they can find job suiting their skills, position, industry, role and location by company name.

* The job Skills recommended application is an example of a search where documents are bulky because of the content in candidate resumes.
* The search provide over the candidate database is required to have huge set of fields to search.
* The current problem recruitment is done manually, most available jobs in Nigeria can only be applied at the agency can be done for which job seeker shave to go to the agency check the available jobs at the agency.

Solution:

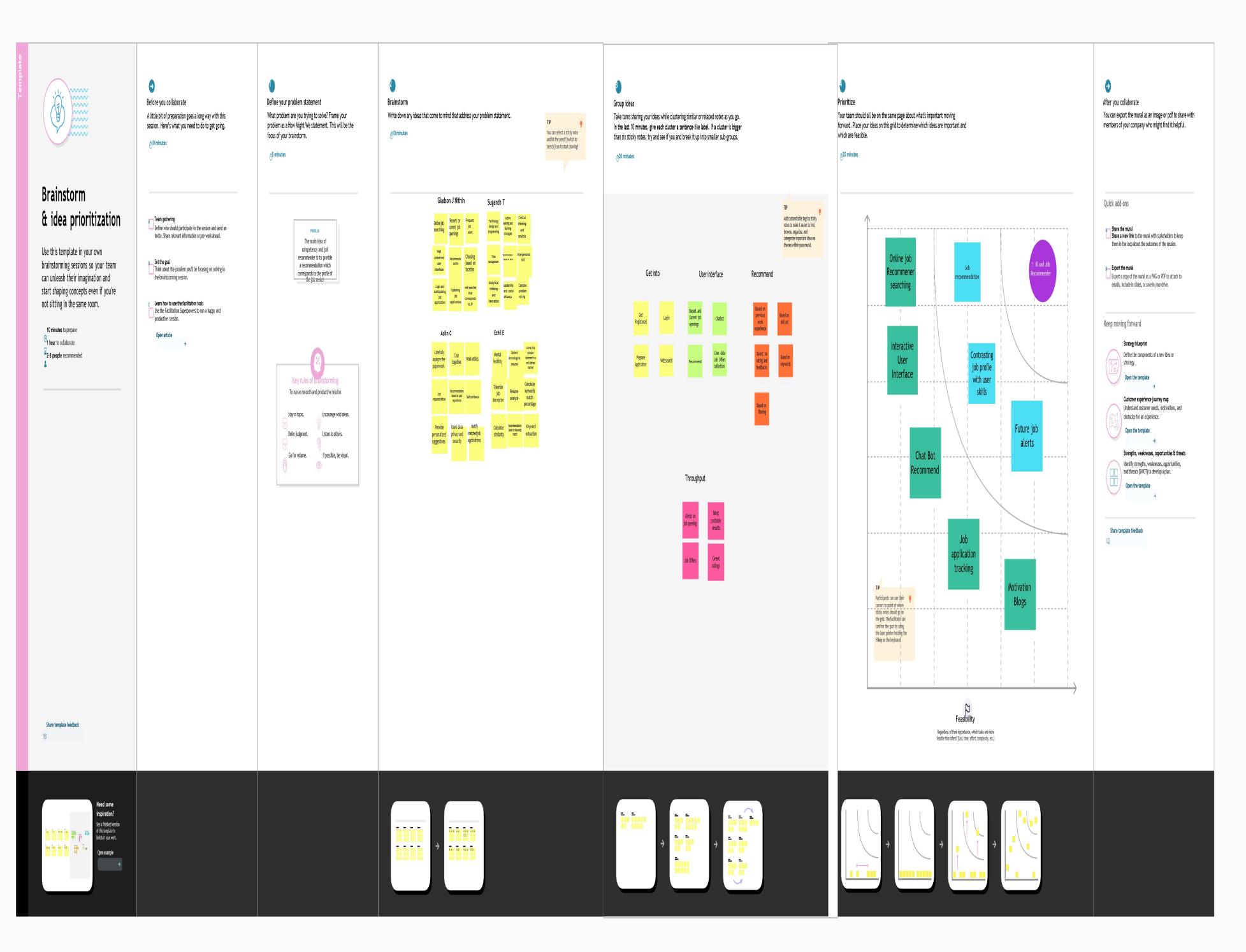
* “The purpose of job-oriented application to help both the job seekers and recruiters find the right organization or the employers.”



## **IDEATION& PROPOSED SOLUTION**

### **Empathy Map Canvas**

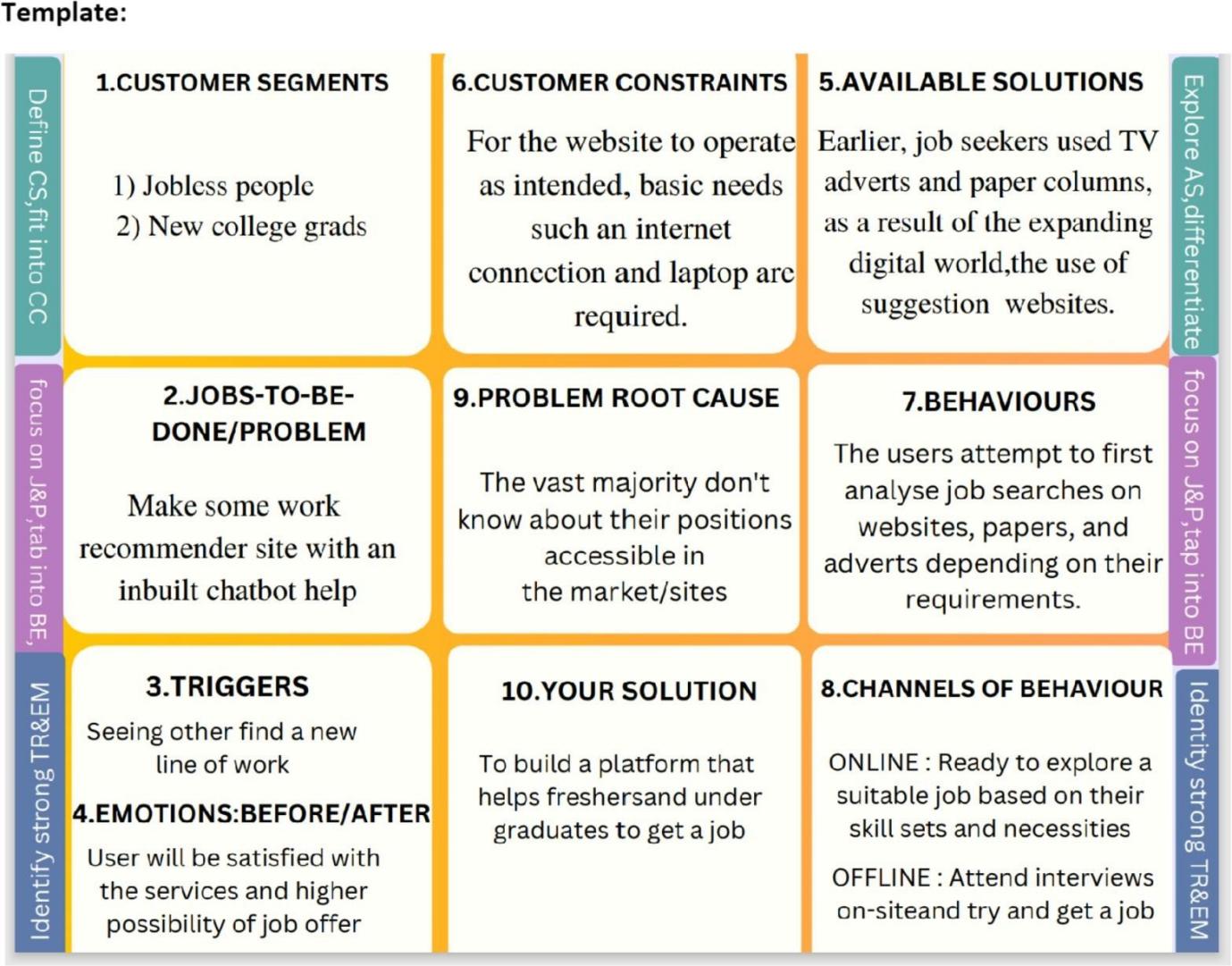
## **Ideation & Brainstorming**



## **Proposed Solution**

|  |  |  |
| --- | --- | --- |
| **S.NO.** | **PARAMETER** | **DESCRIPTION** |
| 1. | Problem Statement (Problem to be solved) | Nowadays a lot of students have great skills but unable to get a desired/appropriate job, so an end-to-end web application can be created which is capable of displaying current job openings based on user skill set making it easier to hire and get hired. |
| 2. | Idea / Solution description | To develop an end-to-end web application which in default have a lot of current job openings through job search API out of which appropriate job will be recommended based on user skill set. At the same time students can develop their skills side by side with various courses and webinars offered by reputed organization. In addition to this a smart chat bot will be available for 24\*7 which can help users in finding the right job. |
| 3. | Novelty / Uniqueness | Though we have a lot of job searching applications, this one is unique because,   * We have a smart chatbot built with IBM Watson * Our platform not only helps in getting job but also helps in developing skills to get right job * Here you can save/bookmark jobs for later use and also turn on notification for company specific job alerts * Add media files to your profile to showcase your achievements * It is made responsive to all screen sizes |
| 4. | Social Impact / Customer Satisfaction | Students will be benefited as they will get to know which job suits them based on their skill set and therefore Lack of Unemployment can be reduced. |
| 5. | Business Model (Revenue Model) | We can provide the application for job seekers in a subscription based and we can share the profiles with companies and generate the revenue by providing them best profiles. |
| 6. | Scalability of the Solution | Data can be scaled up and scaled down according to number of current job openings available. |

# Problem solution fit



# REQUIREMENT ANALYSIS

# Function Requirement

**Software Required:**

Python, Flask, Docker

**Non-Function Requirement**

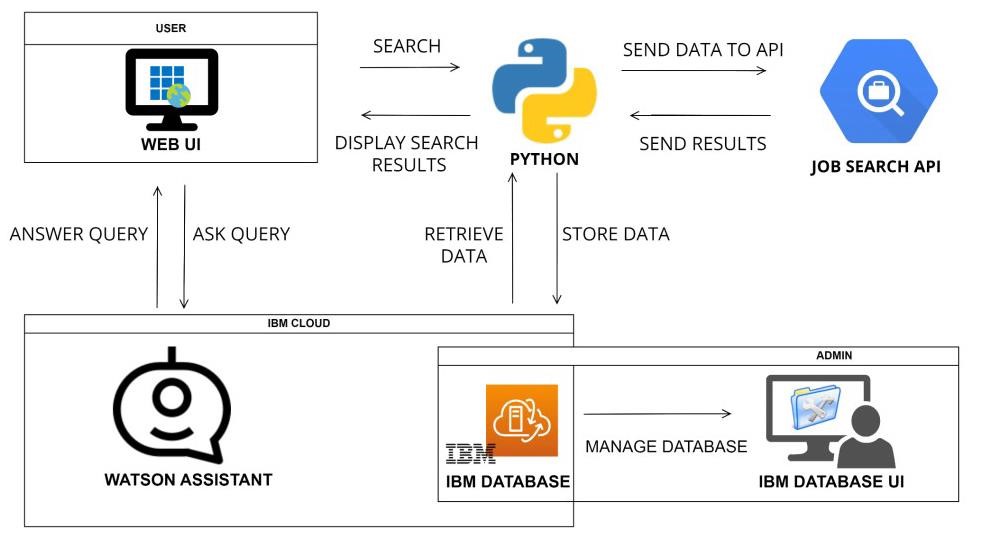
**System Required:**

8GB RAM, Intel Core i3, OS- Windows/Linux/MAC

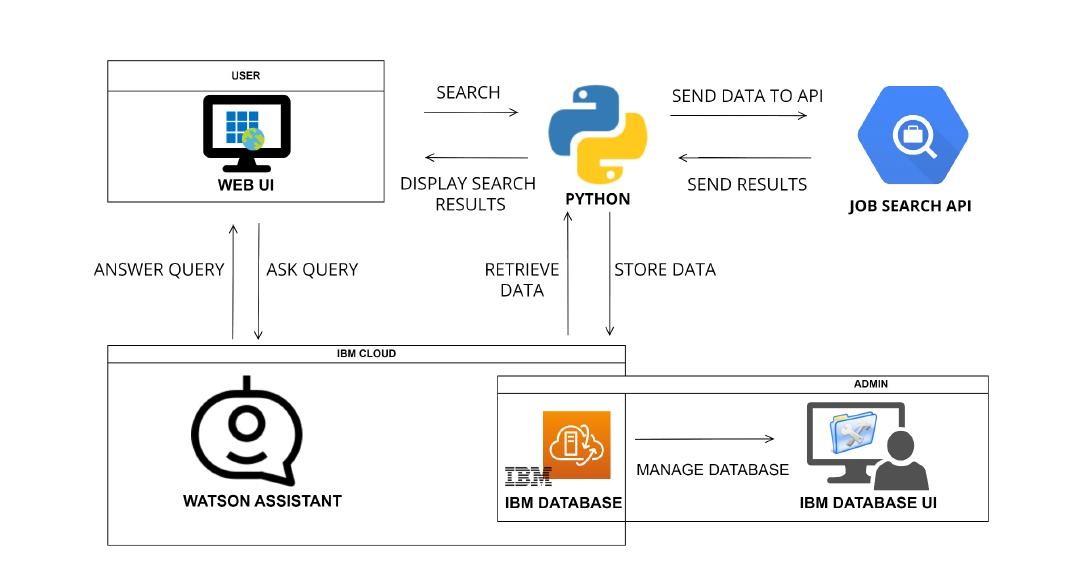
Laptop or Desktop

**PROJECT DESIGN**

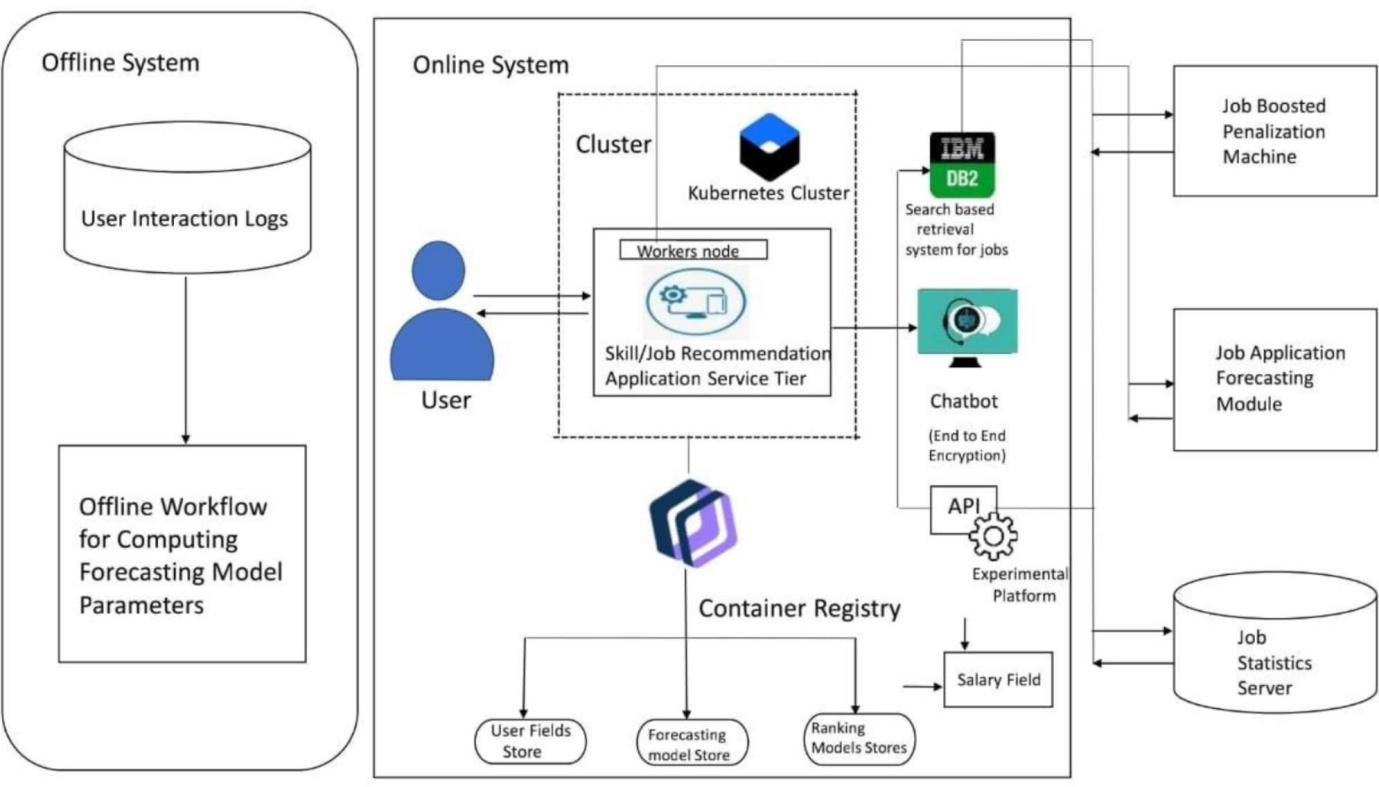
Data Flow Diagrams



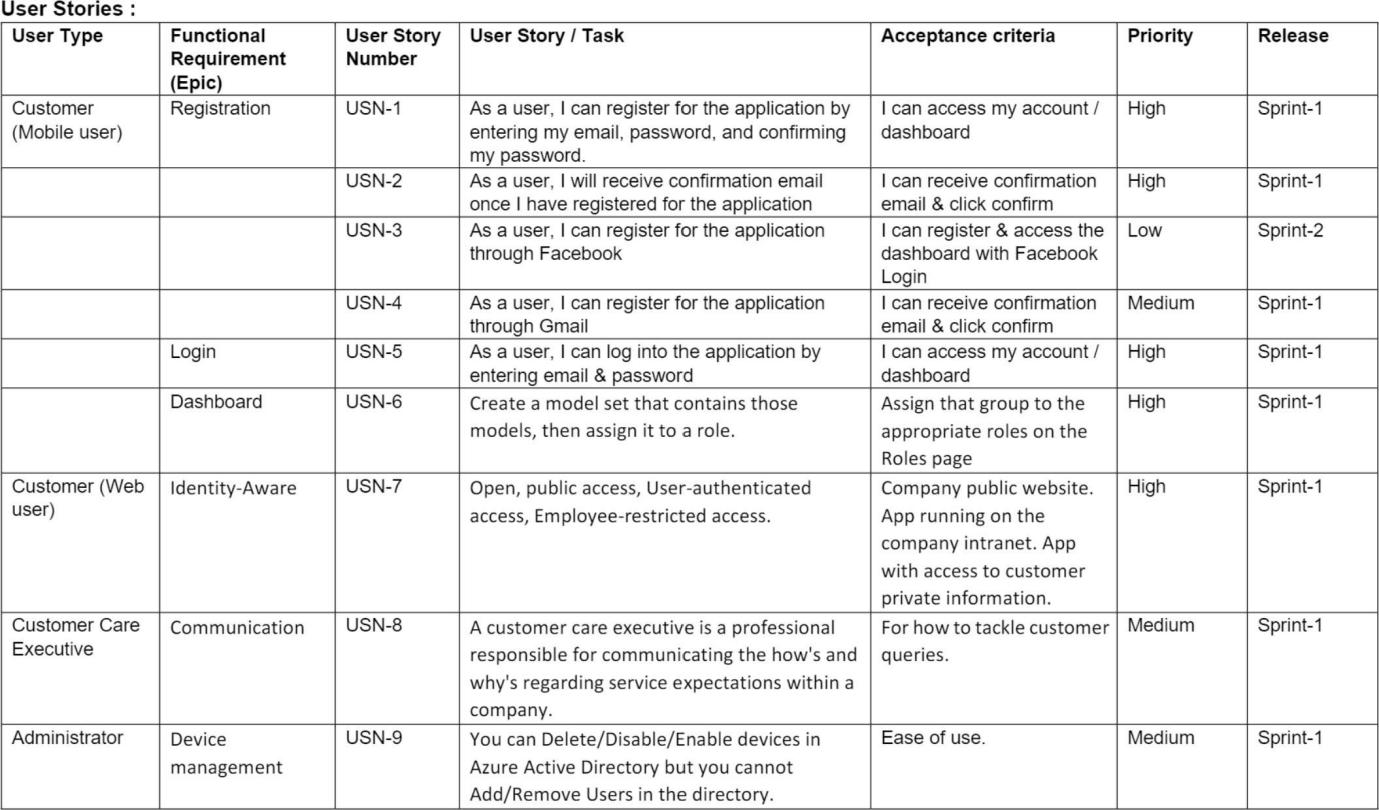
**Solution & Technical Architecture**



## **TECHNICAL ARCHITECTURE**



### **User Stories**



# PROJECT PLANNING & SCHEDULING

## **Sprint Planning & Estimation**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Functional**  **Requirement (Epic)** | **User**  **Story**  **Number** | **User Story / Task** | **Priority** | **Acceptance criteria** | **Team Members** |
| Sprint-1 | UI Design | USN-1 | As a user, I can see and experience an awesome user interface in the website | Medium | Better Impression about a website | NiGladson J Nithin |
| Sprint-1 | Registration | USN-2 | As a user, I can register for the application by entering my email, password, and confirming my password. | High | I can access my account *I* dashboard | Glasdon J Nithin |
| Sprint-1 |  | USN-3 | As a user, I will receive confirmation email once I have registered for the application | High | I can receive confirmation email & click confirm | Gladson J Nithin |
| Sprint-1 |  | USN-4 | As a user, I can register for the application through Facebook | Low | I can register & access the dashboard with Facebook Login | Gladson J Nithin |
| Sprint-1 |  | USN-5 | As a user, I can register for the application through Gmail | Medium | I can receive confirmation email & click confirm | Gladson J Nithin |
| Sprint-1 | Login | USN-6 | As a user, I can log into the application by entering email & password | High | I can access my account *I* dashboard | Gladson J Nithin |
| Sprint-! | Flask | USN-7 | As a user, I can access the website in a second | High | I can access my account *I* dashboard | Gladson J Nithin |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Functional**  **Requirement (Epic)** | **User**  **Story**  **Number** | **User Story / Task** | **Priority** | **Acceptance criteria** | **Team Members** |
| Sprint-1 | Dashboard | USN-8 | As a user, If I Logged in correctly, I can view my dashboard and I can navigate to any pages which are already listed there. | High | I can access all the pages/ dashboard | Gladson J Nithin |
|  |  |  | Submission Of Sprint-1 |  |  |  |
| Sprint-2 | User Profile | USN-9 | As a user, I can view and update my details | Medium | I can modify my details/data | Aslin. C |
| Sprint-2 | Database | USN-10 | As a user, I can store my details and data in the website w | Medium | I can store my data | Aslin. C |
| Sprint-2 | Cloud Storage | USN-11 | As a user, I can upload my photo, resume and much more in the website. | Medium | I can Upload my documents and details | Aslin. C |
| Sprint-2 | Chatbot | USN-12 | As a user, I can ask the Chatbot about latest job openings, which will help me and show the recent job openings based on my profile | High | I can know the recent job openings | Aslin. C |
| Sprint-2 | Identity-Aware | USN-13 | As a User, I can access my account by entering by correct login credentials. My user credentials is only displayed to me. | High | I can have my account safely | Aslin. C |
|  |  |  | Submission of Sprint-2 |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Functional**  **Requirement (Epic)** | **User**  **Story**  **Number** | **User Story / Task** | **Priority** | **Acceptance criteria** | **Team Members** |
| Sprint-3 | Sendgrid service | USN-14 | As a user, I can get a notification or mail about a job opening with the help of sendgrid service. | Medium | I can get a notification in a second. | Suganth. T |
| Sprint-3 | Learning Resource | USN-15 | As a user, I can learn the course and I will attain the skills which will be useful for developing my technical skills. | High | I can gain the knowledge and skills | Suganth. T |
| Sprint-3 | Docker | USN-16 | As a user, I can access the website in any device | High | I can access my account in any device | Suganth. T |
| Sprint-3 | Kubernates | USN-17 | As a user, I can access the website in any device | High | I can access my account in any device | Suganth. T |
| Sprint-3 | Deployment in cloud | USN-18 | As a user, I can access the website in any device | High | I can access my account in any device | Suganth. T |
| Sprint-3 | Technical support | USN-19 | As a user, I can get a customer care support from the website which will solve my queries. | Medium | I can tackle my problem & queries. | Suganth. T |
|  |  |  | Submission of Sprint-3 |  |  |  |
| Sprint-4 | Unit Testing | USN-15 | As a user, I can access the website without any interruption | High | I can access the website without any interruption | Ezhil Nirmals Raj. E |
| Sprint-4 | Integration testing | USN-16 | As a user, I can access the website without any interruption | High | I can access the website without any interruption | Ezhil Nirmals Raj. E |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Functional**  **Requirement (Epic)** | **User**  **Story**  **Number** | **User Story / Task** | **Priority** | **Acceptance criteria** | **Team Members** |
| Sprint-4 | System  testing | USN-17 | As a user, I can access the website without any interruption | High | I can access the website without any interruption | Ezhil Nirmals Raj. E |
| Sprint-4 | Correction | USN-18 | As a user, I can access the website without any interruption | High | I can access the website without any interruption | Ezhil Nirmals Raj. E |
| Sprint-4 | Acceptance testing | USN-19 | As a user, I can access the website without any interruption | High | I can access the website without any interruption | Ezhil Nirmals Raj. E |
|  |  |  | Submission of Sprint-4 |  |  |  |

**Sprint Delivery planning:**

Project Tracker, Velocity & Burndown Chart:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint | Total Story Points | Duration | Sprint Start Date | Sprint End Date (Planned) | Story Points  Completed (as on  Planned End Date) | Sprint Release Date (Actual) |
| Sprint-1 | 20 | 6 Days | 24 Oct 2022 | 29 Oct 2022 | 20 | 06 Nov 2022 |
| Sprint-2 | 20 | 6 Days | 31 Oct 2022 | 05 Nov 2022 | 20 | 10 Nov 2022 |
| Sprint-3 | 20 | 6 Days | 07 Nov 2022 | 12 Nov 2022 | 20 | 15 Nov 2022 |
| Sprint-4 | 20 | 6 Days | 14 Nov 2022 | 19 Nov 2022 | 20 | 19 Nov 2022 |

**Velocity:**

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

# AV sprint duration 20 2 velocity 10

**Burndown Chart:**

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.

**Burndown**

**Cha**

**rt**

**30**

**25**

**20**

**15**

**10**

**5**

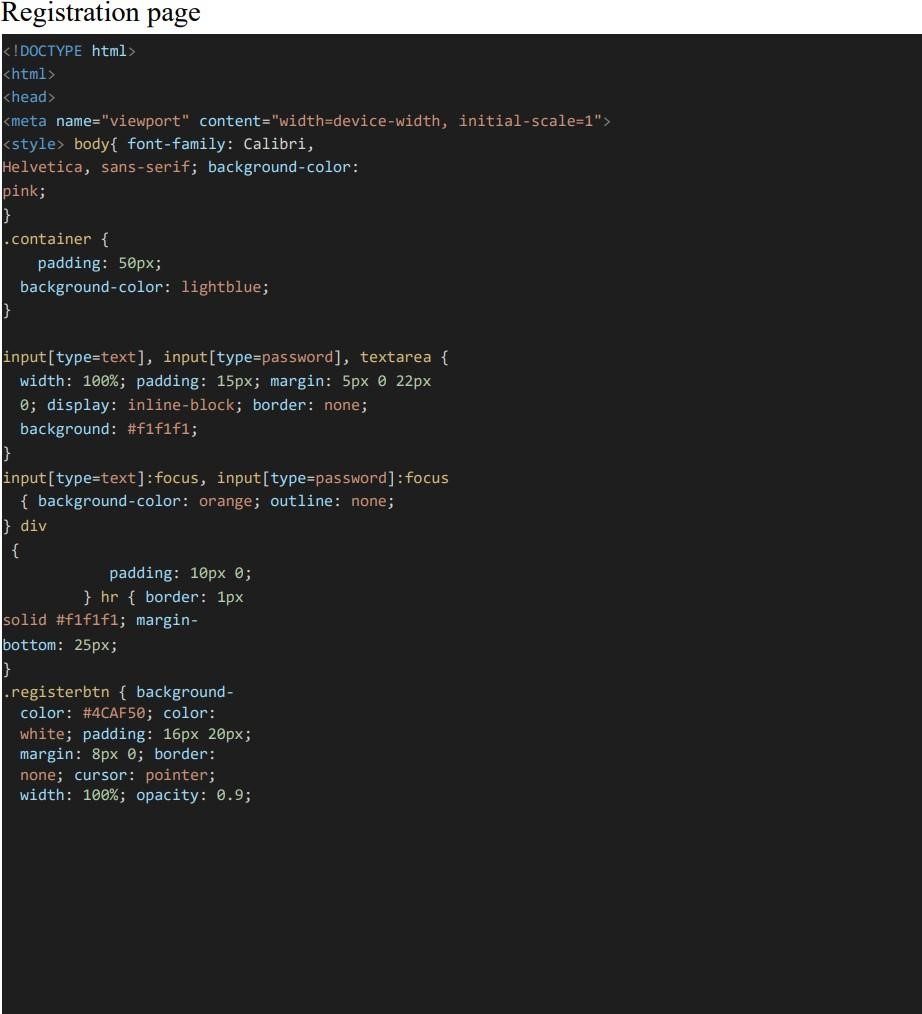
**0**

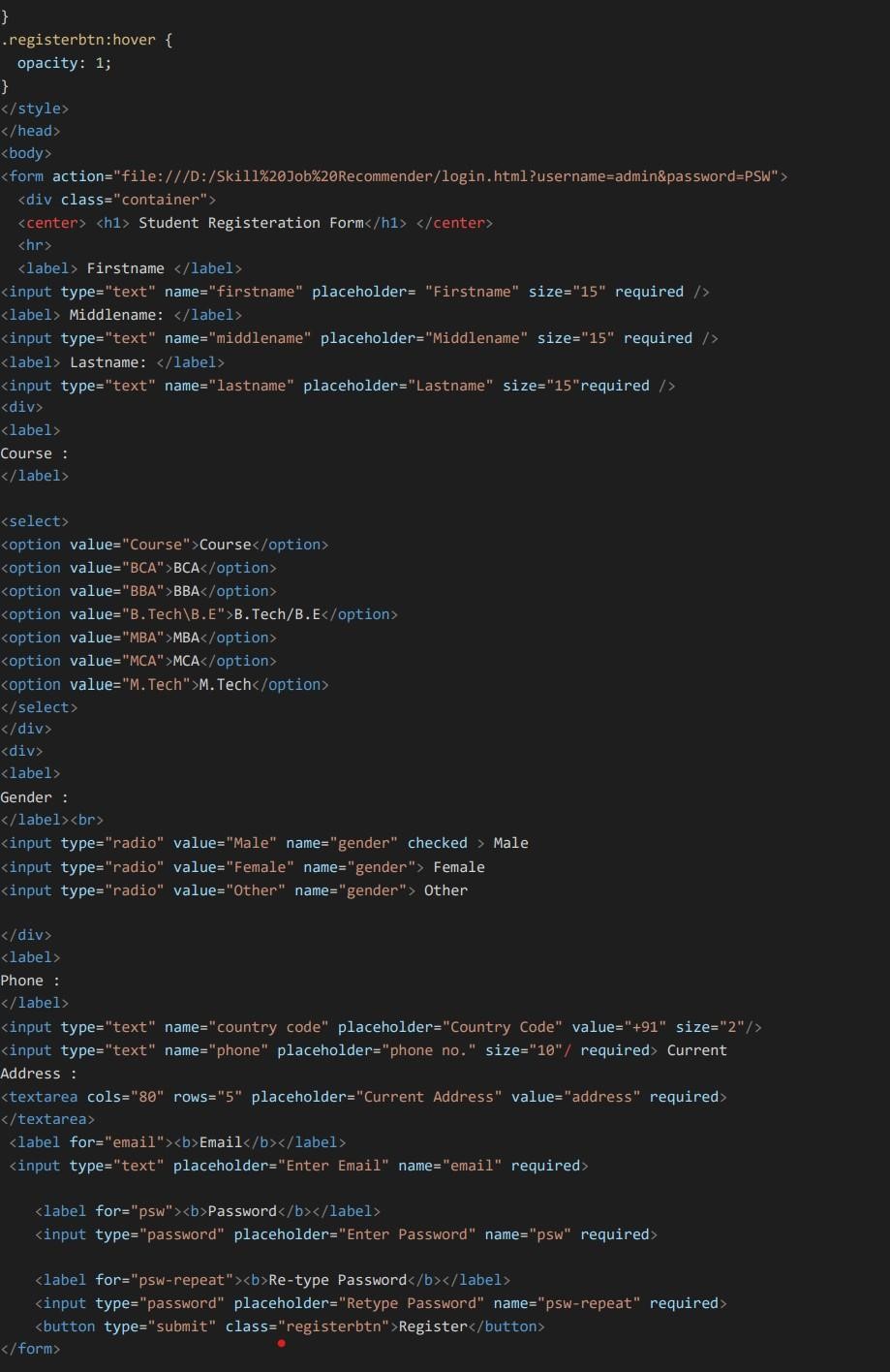
**Sprint 1 Sprint 2 Sprint 3 Sprin t 4**

## Pl anned Act ua l

**7****. CODING & SOLUTIONING**

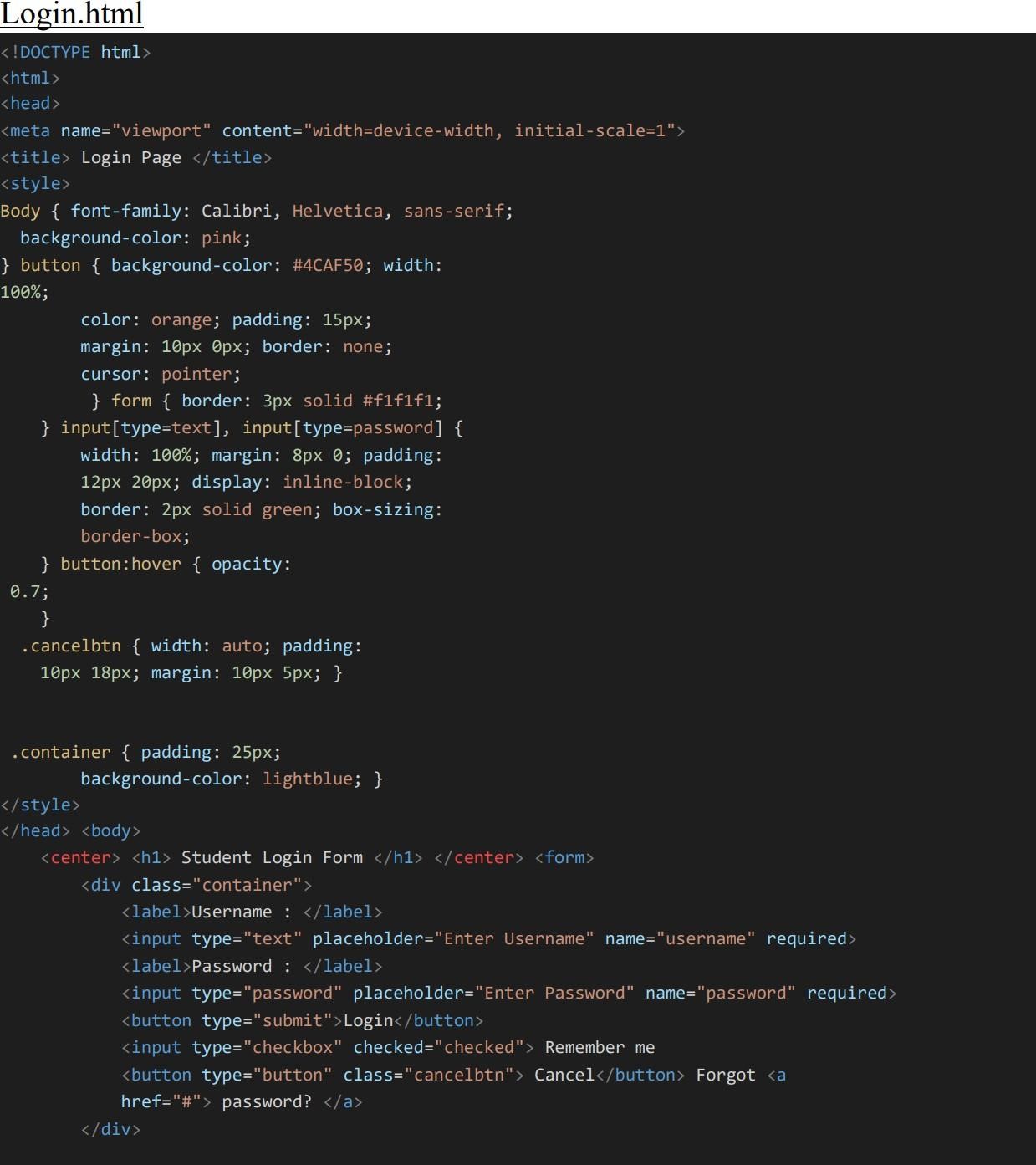
Feature 1







# 





# Feature 2

import { useToast } from "@chakra-ui/react"; import React, { useContext } from "react"; import { Link, useNavigate } from "react-router-dom"; import { AppContext } from "../context/AppContext";

const Navbar = () => {

const navigate = useNavigate();

const toast = useToast();

const { user, setUser, setSkills } = useContext(AppContext);

const logout = () => {

setUser(null);

setSkills([]);

toast({

title: "Logged out successfully!", status: "info", duration: 3000, isClosable: true, variant: "left-accent", position: "top",

});

localStorage.removeItem("user");

navigate("/");

};

return (

<div className="navbar bg-base-100 border-b-2">

<div className="flex-1">

<Link className="btn btn-ghost normal-case text-xl" to={user ? "/dashboard" : "/"}

>

F-ing Jobs

</Link>

</div>

{user && (

<div className="flex-none gap-2">

<div className="dropdown dropdown-end">

<label tabIndex={0} className="btn btn-ghost btn-circle avatar ">

<div className="w-10 rounded-full ring ring-opacity-50 ring-purple-

700">

<img src="https://placeimg.com/80/80/people" /> </div>

</label>

<ul tabIndex={0} className="mt-3 p-2 shadow menu menu-compact dropdown-content

bg-base-100 rounded-box w-52"

>

<li>

<a className="justify-between" onClick={() => navigate("/profile")}

>

Profile

</a> </li>

<li>

<a onClick={logout}>Logout</a> </li>

</ul>

</div>

</div>

)}

</div>

);

}

**Database Schema(if Applicable) :**

# using SendGrid's Python Library

# https://github.com/sendgrid/sendgrid-python import os from sendgrid import SendGridAPIClient from sendgrid.helpers.mail import Mail

# from\_address we pass to our Mail object, edit with your name FROM\_EMAIL = 'Your\_Name@SendGridTest.com'

def SendEmail(to\_email):

""" Send an email to the provided email addresses

:param to\_email = email to be sent to

:returns API response code

:raises Exception e: raises an exception """ message = Mail( from\_email=FROM\_EMAIL, to\_emails=to\_email, subject='A Test from SendGrid!', html\_content='<strong>Hello there from SendGrid your URL is: ' +

<a href=''https://github.com/cyberjive''>right here!</a></strong>') try:

sg = SendGridAPIClient(os.environ.get('SENDGRID\_API\_KEY')) response = sg.send(message) code, body, headers = response.status\_code, response.body,

response.headers print(f"Response Code: {code} ") print(f"Response Body: {body} ") print(f"Response Headers: {headers} ") print("Message Sent!")

except Exception as e:

print("Error: {0}".format(e)) return str(response.status\_code)

if name == " main ":

SendEmail(to\_email=input("Email address to send to? "))

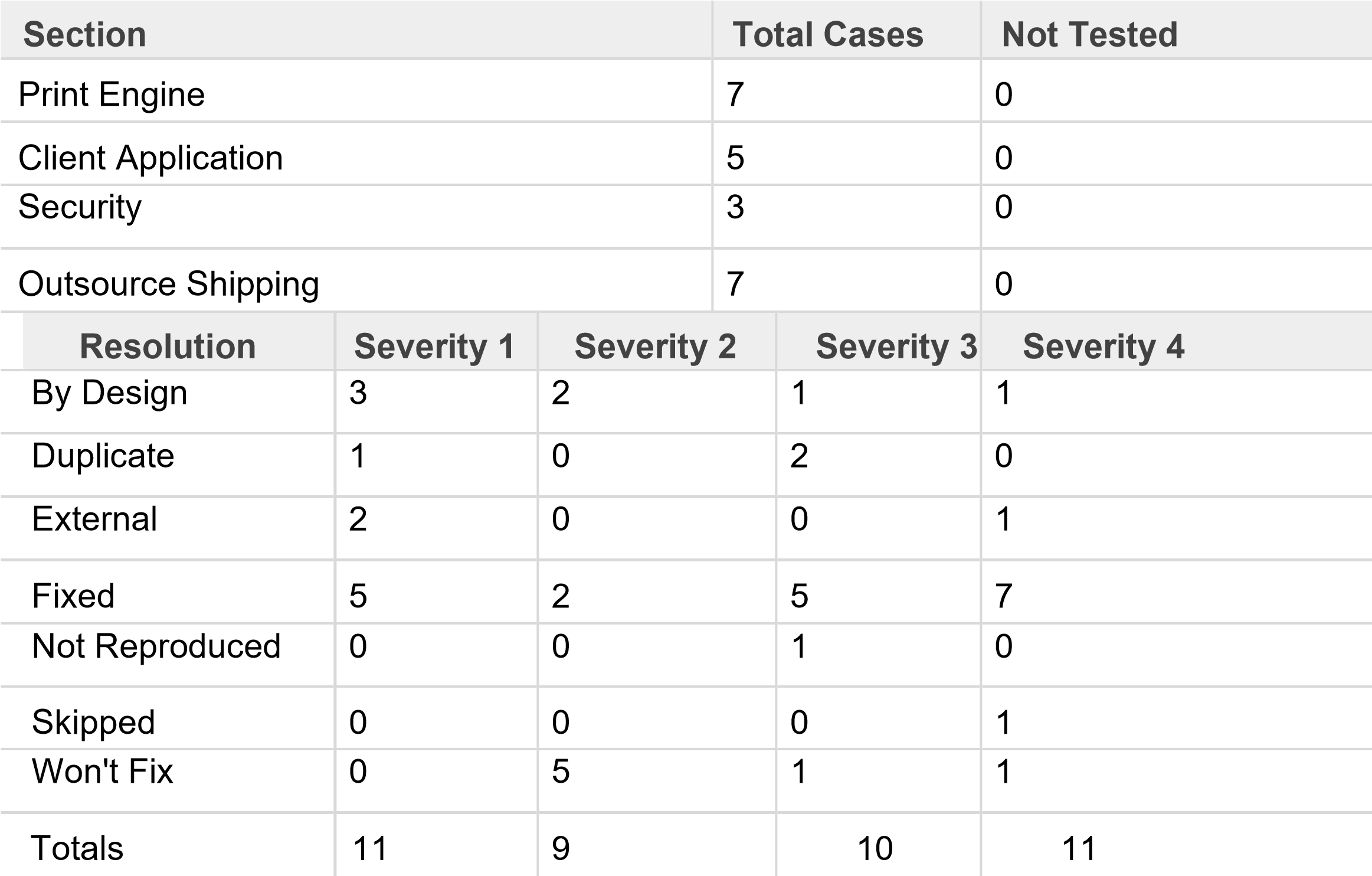
# 8. TESTING

**1. Test Case Analysis**

This report shows the number of test cases that have passed, failed, and untested

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Exception Reporting | 6 | 0 | 0 | 6 |
| Final Report Output | 3 | 0 | 0 | 3 |
| Version Control | 2 | 0 | 0 | 2 |

**8.2 User Acceptance Testing**

The purpose of this document is to briefly explain the test coverage and open issues of the Skills/Job Recommender. Application project at the time of the release to User Acceptance Testing (UAT).

# 9. RESULTS

The project has been completed as we expected. We ensured that Database was designed and well connected to our project. The Expected results were gotten.

# 10. ADVANTAGES & DISADVANTAGES

**ADVANTAGES:**

* Person who looks for a job can easily find a suitable job based on their skill set.
* Person can check their eligibility by attending eligibility test.
* Most of the Recruiters find the suitable person based on the scores they have gotten in the eligibility.

**DISADVANTAGES**

➢Person Job May get technical difficulty while taking the eligibility

Job seeker may have trouble to contact recruiters directly

# 11. CONCLUSION

The application has been developed to make job search easier .The application that we have developed is user friendly .User can find a job based on their skillset in the short period of time. The jobseeker certainly get benefit by using this application. In the addition, Chatbot Has been implemented with the help of IBM whatson. The chatbot helps jobseeker and organization when they experience the difficulties.

# 12. FUTURE SCOPE

The linked in the well-known application to find a job and stay connected with professional and organization. The job seekers and organization use linked in to find a job. In the future, There are lots of possibilities to enhance our project similar to linked in

# 13. APPENDIX

**SOURCE CODE:**

**SOURCE CODE**

**init .py**

from dotenv import dotenv\_values

from flask import Flask

from flask\_cors import CORS

import ibm\_db

# Get the environment variables

config = dotenv\_values("backend/.env")

# Connect to db

try:

# conn = 'dd'

conn = ibm\_db.pconnect(

f"DATABASE={config['DB2\_DATABASE']};HOSTNAME={config['DB2\_HOSTNAME']};

PORT={config['DB2\_PORT']};SECURITY=SSL; SSLServerCertificate=backend/

DigiCertGlobalRootCA.crt;UID={config['DB2\_USERNAME']};

PWD={config['DB2\_PASSWORD']}", '', '')

print("Connected to IBM\_DB2 successfully!!")

print(conn)

except:

print("Failed to connect to Database!")

def create\_app():

# Tell flask to use the build directory of react to serve static content

app = Flask( name ,static\_folder='../build', static\_url\_path='/')

CORS(app)

# Set the secret key for flask

app.config['SECRET\_KEY'] = config['APP\_SECRET']

# Import and register auth\_router

from .auth\_router import auth

app.register\_blueprint(auth, url\_prefix='/api/auth')from .files\_router import files

app.register\_blueprint(files, url\_prefix='/api/files')

from .user\_router import user

app.register\_blueprint(user, url\_prefix='/api/user')

# In production serve the index.html page at root

@app.route("/")

def home():

return app.send\_static\_file('index.html')

return app

**auth\_middleware.py**

from functools import wraps

import jwt

from flask import request

from backend import conn, config

import ibm\_db

# Middleware function that checks for JWT token in header

# All routes that have the @token\_required decorator will be protected

def token\_required(f):

@wraps(f)

def decorated(\*args, \*\*kwargs):

token = None

if "Authorization" in request.headers:

token = request.headers["Authorization"].split(" ")[1]

if not token:

return {

"error": "Unauthorized"

}, 401

try:

# Get the user's email from the decoded token

data = jwt.decode(

token, config["APP\_SECRET"], algorithms=["HS256"])

# Retreive user's info from the database

sql = f"select \* from users where email='{data['email']}'"

stmt = ibm\_db.prepare(conn, sql)

ibm\_db.execute(stmt)

current\_user = ibm\_db.fetch\_assoc(stmt)

# If user does not exist throw error.

if current\_user is None:

return {

"error": "Unauthorized"

}, 401

except Exception as e:

return {

"error": str(e)

}, 500

# Pass the authorized user in function args.

return f(current\_user, \*args, \*\*kwargs)

return decorated

**auth\_router.py**

from flask import Blueprint, jsonify, request

from backend import conn, config

import bcrypt

import jwt

import ibm\_db

auth = Blueprint("auth",

name )

LOGIN\_FEILDS = ('email', 'password')

SIGNUP\_FEILDS = ('name', 'email', 'phone\_number', 'password')

@auth.route("/login", methods=['POST'])

def login\_user():

# Check if all the required feild are present

for feild in LOGIN\_FEILDS:if not (feild in request.json):

return jsonify({"error": f"All feilds are required!"}), 409

email = request.json['email']

password = request.json['password']

sql = f"select \* from users where email='{email}'"

stmt = ibm\_db.prepare(conn, sql)

ibm\_db.execute(stmt)

user = ibm\_db.fetch\_assoc(stmt)

if not user:

return jsonify({"error": "Invalid credentials!"}), 401

if bcrypt.checkpw(password.encode('utf-8'),

user["PASSWORD"].encode('utf-8')):

token = jwt.encode(

{"email": email},

config["APP\_SECRET"],

algorithm="HS256"

)

return jsonify({"name": user["NAME"], "email": email, "phone\_number":

user["PHONE\_NUMBER"], "token": token}), 200

else:

return jsonify({"error": "Invalid credentials!"}), 401

@auth.route("/signup", methods=['POST'])

def register\_user():

# Check if all the required feild are present

for feild in SIGNUP\_FEILDS:

if not (feild in request.json):

return jsonify({"error": f"All feilds are required!"}), 409

email = request.json['email']

phone\_number = request.json['phone\_number']

name = request.json['name']

password = request.json['password']

# Sql stmt to check if email/number is already in use

sql

=

f"select

\*

from

users

where

email='{email}'

or

phone\_number='{phone\_number}'"

stmt = ibm\_db.prepare(conn, sql)

ibm\_db.execute(stmt)

user = ibm\_db.fetch\_assoc(stmt)if user:

return jsonify({"error": f"Email/Phone number is alread in use!"}), 409

# If user does not exist, then create account

hashed\_password = bcrypt.hashpw(

password.encode('utf-8'), bcrypt.gensalt())

sql

=

f"insert

into

users(name,email,phone\_number,password)

values('{name}','{email}','{phone\_number}',?)"

stmt = ibm\_db.prepare(conn, sql)

ibm\_db.bind\_param(stmt, 1, hashed\_password)

ibm\_db.execute(stmt)

token = jwt.encode(

{"email": email},

config["APP\_SECRET"],

algorithm="HS256"

)

return jsonify({"name": name, "email": email, "phone\_number": phone\_number,

"token": token}), 200

**files\_router.py**

from flask import Blueprint

from backend.auth\_middleware import token\_required

import ibm\_boto3

from ibm\_botocore.client import Config, ClientError

from backend import config

cos = ibm\_boto3.resource("s3",

ibm\_api\_key\_id=config["COS\_API\_KEY\_ID"],

ibm\_service\_instance\_id=config["COS\_INSTANCE\_CRN"],

config=Config(signature\_version="oauth"),

endpoint\_url=config["COS\_ENDPOINT"]

)

files = Blueprint("files",

name )

def multi\_part\_upload(bucket\_name, item\_name, file\_path):

try:

print("Starting file transfer for {0} to bucket: {1}\n".format(

item\_name, bucket\_name))# set 5 MB chunks

part\_size = 1024 \* 1024 \* 5

# set threadhold to 15 MB

file\_threshold = 1024 \* 1024 \* 15

# set the transfer threshold and chunk size

transfer\_config = ibm\_boto3.s3.transfer.TransferConfig(

multipart\_threshold=file\_threshold,

multipart\_chunksize=part\_size

)

# the upload\_fileobj method will automatically execute a multi-part upload

# in 5 MB chunks for all files over 15 MB

with open(file\_path, "rb") as file\_data:

cos.Object(bucket\_name, item\_name).upload\_fileobj(

Fileobj=file\_data,

Config=transfer\_config

)

print("Transfer for {0} Complete!\n".format(item\_name))

except ClientError as be:

print("CLIENT ERROR: {0}\n".format(be))

except Exception as e:

print("Unable to complete multi-part upload: {0}".format(e))

@files.route('/avatar', methods=["POST"])

@token\_required

def upload\_profile\_photo(current\_user):

return "hello"

**user\_router.py**

from flask import Blueprint, jsonify, request

from backend import conn

from backend.auth\_middleware import token\_required

import ibm\_db

user = Blueprint("user",

name )@user.route("/skills", methods=["GET", "POST", "DELETE"])

@token\_required

def manage\_skills(current\_user):

# Get user\_id of current user

user\_id = current\_user['USER\_ID']

# Handle GET request

if request.method == 'GET':

skills = []

sql = f"select name from skills where user\_id={user\_id}"

stmt = ibm\_db.prepare(conn, sql)

ibm\_db.execute(stmt)

dict = ibm\_db.fetch\_assoc(stmt)

# Iterate over all the results and append skills to the array

while dict != False:

skills.append(dict['NAME'])

dict = ibm\_db.fetch\_assoc(stmt)

return jsonify({"skills": skills}), 200

# Get the skills from the request

if not ('skills' in request.json):

return jsonify({"error": f"All feilds are required!"}), 409

skills = request.json['skills']

# If no skills are provided then return empty array

if skills == []:

return jsonify({"skills": []}), 200

# Handle POST request

if request.method == "POST":

# Prepare the SQL statement to insert multiple rows

values = ''

for i in range(len(skills)):

if i == 0:

values += 'values'

values += f"('{skills[i]}',{user\_id})"if i != len(skills)-1:

values += ','

sql = f"insert into skills(name,user\_id) {values}"

stmt = ibm\_db.prepare(conn, sql)

status = ibm\_db.execute(stmt)

if status:

return jsonify({"message": "Updated skills successfully!"}), 200

else:

jsonify({"error": "Something went wrong!!"}), 409

# Handle DELETE request

if request.method == 'DELETE':

values = ""

for i in range(len(skills)):

values += f"'{skills[i]}'"

if i != len(skills)-1:

values += ','

sql = f"delete from skills where name in ({values})"

stmt = ibm\_db.prepare(conn, sql)

status = ibm\_db.execute(stmt)

if status:

return jsonify({"message": "Deleted skills successfully!"}), 200

else:

jsonify({"error": "Something went wrong!!"}), 409

**avatar.svg**

<svg width="480" height="480" fill="none"

xmlns="http://www.w3.org/2000/svg"><rect opacity=".1" width="480"

height="480" rx="32" fill="#fff"/><path d="M374.308 240c0 71.691-58.117

129.808-129.808 129.808S114.692 311.691 114.692 240 172.809 110.192

244.5 110.192 374.308 168.309 374.308 240z" fill="#F6F6F6" stroke="#fff"

stroke-width="10.385"/><path fill-rule="evenodd" clip-rule="evenodd"

d="M244.5 256.2c-21.627 0-64.8 10.854-64.8 32.4v16.2h129.6v-16.2c0-

21.546-43.173-32.4-64.8-32.4m0-16.2c17.901 0 32.4-14.499 32.4-32.4 0-

17.901-14.499-32.4-32.4-32.4-17.901 0-32.4 14.499-32.4 32.4 0 17.901 14.499

32.4 32.4 32.4" fill="#35374A" opacity=".3"/></svg>**JobCard.jsx**

import React, { useEffect } from "react";

const JobCard = ({ title, company, description, link }) => {

return (

<div className="max-w-sm flex flex-col rounded overflow-hidden shadow

lg border-2 border-slate-200">

<>

<div className="px-6 py-4">

<div className="font-bold text-xl">{title}</div>

<div className="text mb-2 text-gray-400">{company}</div>

<p className="text-ellipsis overflow-hidden text-gray-800 text-sm">

{description}

</p>

</div>

<div className="px-6 pt-4 pb-2 mt-auto mb-2">

<a

href={link}

target=" blank"

className="bg-transparent hover:bg-purple-400 text-purple-400 font

semibold hover:text-white py-2 mb-0 mt-4 px-4 border border-purple-400

hover:border-transparent rounded"

>

Apply

</a>

</div>

</>

</div>

);};

export default JobCard;

**Login.jsx**

import React, { useContext, useState } from "react";

import { Link, useNavigate } from "react-router-dom";

import { AppContext } from "../context/AppContext";

import { loginUser } from "../proxies/backend\_api";

import { emailRegex } from "../utils/helper";

const Login = () => {

const { setShowAlert, setUser } = useContext(AppContext);

const navigate = useNavigate();

const [inputs, setInputs] = useState({

email: "",

password: "",

});

const [error, setErrors] = useState({

email: "",

password: "",

});

const handleChange = ({ target: { name, value } }) => {

setErrors((prev) => {

return { ...prev, [name]: "" };});

setInputs((prev) => ({ ...prev, [name]: value }));

};

const checkInputErrors = () => {

let status = true;

if (inputs.email.trim() === "" || !emailRegex.test(inputs.email.trim())) {

setErrors((prev) => {

return { ...prev, email: "Enter a valid email" };

});

status = false;

}

if (inputs.password.trim() === "") {

setErrors((prev) => {

return { ...prev, password: "Enter a valid password" };

});

status = false;

}

if (inputs.password.trim().length < 6) {

setErrors((prev) => {

return { ...prev, password: "Minimum 6 characters" };

});

status = false;

}

return status;

};const handleLogin = async () => {

if (checkInputErrors()) {

const data = await loginUser(inputs);

if (data.error) {

setShowAlert({ type: "error", message: data.error, duration: 3000 });

return;

}

setUser(data);

setShowAlert({

type: "success",

message: `Welcome back ${data.name}`,

duration: 3000,

});

localStorage.setItem("user", JSON.stringify(data));

navigate("/dashboard");

}

};

return (

<div className="flex flex-col justify-center items-center gap-10 mt-5">

<div>

<button className="bg-base-300 rounded-box flex flex-row justify-evenly

items-center gap-10 px-10 py-5 w-fit mx-auto">

<span>Sign in with Github</span>

<img src={`github-dark.png`} alt="github" width="14%" />

</button>

<div className="divider max-w-xs">or</div>

<form

onSubmit={(e) => e.preventDefault()}className="card bg-base-300 rounded-box flex flex-col justify-center

items-center gap-5 px-10 py-5 w-fit mx-auto"

>

<div>

<input

value={inputs.email}

type="text"

name="email"

placeholder="email"

className="input input-bordered input-primary w-full"

onChange={handleChange}

/>

{error.email !== "" && (

<p className="text-sm text-red-500 mt-1 font-medium">

{error.email}

</p>

)}

</div>

<div>

<input

value={inputs.password}

type="password"

name="password"

placeholder="password"

className="input input-bordered input-primary w-full"

onChange={handleChange}

/>

{error.password !== "" && (

<p className="text-sm text-red-500 mt-1 font-medium">{error.password}

</p>

)}

</div>

<div className="text-center">

<button

type="submit"

onClick={handleLogin}

className="btn btn-sm btn-primary mb-4"

>

Login

</button>

<p>

Don't have an account?{" "}

<Link className="text-blue-400" to="/signup">

Sign up

</Link>

</p>

</div>

</form>

</div>

</div>

);

};

export default Login;

**Navbar.jsx**

import { useToast } from "@chakra-ui/react";

import React, { useContext } from "react";

import { Link, useNavigate } from "react-router-dom";

import { AppContext } from "../context/AppContext";

const Navbar = () => {

const navigate = useNavigate();

const toast = useToast();

const { user, setUser, setSkills } = useContext(AppContext);

const logout = () => {

setUser(null);

setSkills([]);

toast({

title: "Logged out successfully!",

status: "info",

duration: 3000,

isClosable: true,

variant: "left-accent",

position: "top",

});

localStorage.removeItem("user");navigate("/");

};

return (

<div className="navbar bg-base-100 border-b-2">

<div className="flex-1">

<Link

className="btn btn-ghost normal-case text-xl"

to={user ? "/dashboard" : "/"}

>

F-ing Jobs

</Link>

</div>

{user && (

<div className="flex-none gap-2">

<div className="dropdown dropdown-end">

<label tabIndex={0} className="btn btn-ghost btn-circle avatar ">

<div className="w-10 rounded-full ring ring-opacity-50 ring-purple-

700">

<img src="https://placeimg.com/80/80/people" />

</div>

</label>

<ul

tabIndex={0}

className="mt-3 p-2 shadow menu menu-compact dropdown-content

bg-base-100 rounded-box w-52"

>

<li>

<aclassName="justify-between"

onClick={() => navigate("/profile")}

>

Profile

</a>

</li>

<li>

<a onClick={logout}>Logout</a>

</li>

</ul>

</div>

</div>

)}

</div>

);

};

export default Navbar;

**SearchBar.jsx**

import React from "react";

import { BsSearch } from "react-icons/bs";

const SearchBar = ({ setquery, onClick }) => {

const handlesubmit = (e) => {

e.preventDefault();

onClick();

};return (

<form className="flex items-center" onSubmit={handlesubmit}>

<label htmlFor="simple-search" className="sr-only">

Search

</label>

<div className="relative w-full">

<div className="flex absolute inset-y-0 left-0 items-center pl-3 pointer

events-none">

<BsSearch />

</div>

<input

onChange={(e) => setquery(e.target.value)}

name="search"

type="text"

id="simple-search"

className="bg-gray-50 border border-gray-300 text-gray-900 text-sm

rounded-lg focus:ring-blue-500 focus:border-blue-500 block w-full pl-10 p-2.5

dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400 dark:text

white dark:focus:ring-blue-500 dark:focus:border-blue-500"

placeholder="Search"

required=""

/>

</div>

<button

type="submit"

className="p-2.5 ml-2 text-sm font-medium text-white bg-purple-700

rounded-lg border border-purple-700 hover:bg-purple-800 focus:ring-4

focus:outline-none focus:ring-purple-300"

>

<BsSearch /><span className="sr-only">Search</span>

</button>

</form>

);

};

export default SearchBar;

**Signup.jsx**

import React, { useContext, useEffect, useState } from "react";

import { useNavigate } from "react-router-dom";

import { AppContext } from "../context/AppContext";

import { registerUser } from "../proxies/backend\_api";

import { emailRegex } from "../utils/helper";

const SignUp = () => {

const { setUser } = useContext(AppContext);

const navigate = useNavigate();

const [inputs, setInputs] = useState({

name: "",

email: "",

phone\_number: "",

password: "",

confirm\_password: "",

});

const [error, setErrors] = useState({name: "",

email: "",

phone\_number: "",

password: "",

confirm\_password: "",

});

const handleChange = ({ target: { name, value } }) => {

setErrors((prev) => {

return { ...prev, [name]: "" };

});

setInputs((prev) => ({ ...prev, [name]: value }));

};

const checkInputErrors = () => {

let status = true;

if (inputs.email.trim() === "" || !emailRegex.test(inputs.email.trim())) {

setErrors((prev) => {

return { ...prev, email: "Enter a valid email" };

});

status = false;

}

if (inputs.name.trim() === "") {

setErrors((prev) => {

return { ...prev, name: "Enter a valid name" };

});

status = false;

}if (inputs.phone\_number.trim() === "") {

setErrors((prev) => {

return { ...prev, phone\_number: "Enter a valid phone number" };

});

status = false;

}

if (inputs.confirm\_password.trim() === "") {

setErrors((prev) => {

return { ...prev, confirm\_password: "Enter a valid password" };

});

status = false;

}

if (inputs.password.trim() === "") {

setErrors((prev) => {

return { ...prev, password: "Enter a valid password" };

});

status = false;

}

if (inputs.password.trim().length < 6) {

setErrors((prev) => {

return { ...prev, password: "Minimum 6 characters" };

});

status = false;

}if (inputs.password.trim() !== inputs.confirm\_password.trim()) {

setErrors((prev) => {

return { ...prev, confirmPassword: "Password don't match" };

});

status = false;

}

return status;

};

const handleSignUp = async () => {

if (checkInputErrors()) {

const data = await registerUser(inputs);

if (data.error) {

toast({

title: data.error,

status: "error",

duration: 3000,

isClosable: true,

variant: "left-accent",

position: "top",

});

return;

}

setUser(data);

toast({

title: `Your journey starts here ${data.name}`,

status: "success",

duration: 3000,

isClosable: true,variant: "left-accent",

position: "top",

});

localStorage.setItem("user", JSON.stringify(data));

navigate("/profile");

}

};

return (

<>

<div>

<button className="bg-base-300 rounded-box flex flex-row justify-evenly

items-center gap-10 px-10 py-5 w-fit mx-auto">

<span>Sign in with Github</span>

<img src={`github-dark.png`} alt="github" width="14%" />

</button>

<div className="divider max-w-xs">or</div>

<div className="card bg-base-300 rounded-box flex flex-col justify-center

items-center gap-3 px-10 py-5 w-fit mx-auto">

<div>

<input

value={inputs.name}

type="text"

name="name"

placeholder="name"

className="input input-bordered input-primary w-full"

onChange={handleChange}

/>

{error.name !== "" && (<p className="text-sm text-red-500 font-medium">{error.name}</p>

)}

</div>

<div>

<input

value={inputs.email}

type="text"

name="email"

placeholder="email"

className="input input-bordered input-primary w-full"

onChange={handleChange}

/>

{error.email !== "" && (

<p className="text-sm text-red-500 font-medium">{error.email}</p>

)}

</div>

<div>

<input

value={inputs.phone\_number}

type="text"

name="phone\_number"

placeholder="phone number"

className="input input-bordered input-primary w-full"

onChange={handleChange}

/>

{error.phone\_number !== "" && (

<p className="text-sm text-red-500 font-medium">

{error.phone\_number}

</p>)}

</div>

<div>

<input

value={inputs.password}

type="password"

name="password"

placeholder="password"

className="input input-bordered input-primary w-full"

onChange={handleChange}

/>

{error.password !== "" && (

<p className="text-sm text-red-500 font-medium">

{error.password}

</p>

)}

</div>

<div>

<input

value={inputs.confirm\_password}

type="password"

name="confirm\_password"

placeholder="confirm password"

className="input input-bordered input-primary w-full"

onChange={handleChange}

/>

{error.confirm\_password !== "" && (

<p className="text-sm text-red-500 font-medium">

{error.confirm\_password}</p>

)}

</div>

<div className="text-center">

<button

onClick={handleSignUp}

className="btn btn-sm btn-primary mb-4"

>

Sign Up

</button>

</div>

</div>

</div>

</>

);

};

export default SignUp;

**Skill.jsx**

import React, { useEffect, useState } from "react";

const Skill = ({ skill, setSelectedSkills, disabled }) => {

const [isSelected, setIsSelected] = useState(false);

useEffect(() => {

if (isSelected) {

setSelectedSkills((prev) => [...prev, skill]);

} else {setSelectedSkills((prev) => prev.filter((item) => item !== skill));

}

}, [isSelected]);

return (

<li className="hover:text-white flex gap-1 items-center justify-between p-1

rounded-sm">

{skill}

<button

disabled={disabled}

onClick={() => setIsSelected(!isSelected)}

className={`cursor-pointer border-2 ${

!isSelected ? "border-green-500" : "border-red-400"

} p-1 rounded-lg`}

>

{!isSelected ? "Add" : "Remove"}

</button>

</li>

);

};

export default Skill;

**AppContext.jsx**

import { createContext, useEffect, useState } from "react";

import { useNavigate } from "react-router-dom";

export const AppContext = createContext();export const AppProvider = ({ children }) => {

const navigate = useNavigate();

const [skills, setSkills] = useState([]);

const [user, setUser] = useState(null);

useEffect(() => {

let temp\_user = JSON.parse(localStorage.getItem("user"));

if (!temp\_user) {

navigate("/");

} else {

setUser(temp\_user);

}

}, []);

return (

<AppContext.Provider value={{ user, setUser,skills, setSkills }}>

{children}

</AppContext.Provider>

);

};

**backend\_api.js**

import { BASE\_URL } from "../utils/helper";

export const loginUser = async (inputs) => {

try {

const response = await fetch(`${BASE\_URL}/auth/login`, {method: "POST",

body: JSON.stringify(inputs),

headers: {

"Content-Type": "application/json",

},

});

const data = await response.json();

return data;

} catch (error) {

console.error(error);

}

};

export const registerUser = async (inputs) => {

try {

const response = await fetch(`${BASE\_URL}/auth/signup`, {

method: "POST",

body: JSON.stringify(inputs),

headers: {

"Content-Type": "application/json",

},

});

const data = await response.json();

return data;

} catch (error) {

console.error(error);

}

};

**Profile.jsx**

import {

Progress,

SkeletonCircle,

SkeletonText,

Spinner,

useToast,

} from "@chakra-ui/react";

import React, { useContext, useEffect, useState } from "react";

import { AiOutlineClose } from "react-icons/ai";

import { BsLinkedin } from "react-icons/bs";

import { GoMarkGithub } from "react-icons/go";

import { MdDeleteForever } from "react-icons/md";

import { RiEdit2Fill } from "react-icons/ri";

import { TfiTwitterAlt } from "react-icons/tfi";

import { VscAdd } from "react-icons/vsc";

import { AppContext } from "../context/AppContext";

import {

getUserSkills,

removeUserSkills,

saveUserSkills,

updateUserDetails,

} from "../proxies/backend\_api";

const Profile = () => {

const toast = useToast();

const { user, setUser,skills, setSkills } = useContext(AppContext);const [addSkill, setAddSkill] = useState("");

const [newSkills, setNewSkills] = useState([]);

const [removedSkills,setRemovedSkills] = useState([]);

const [isEditingEnabled, setIsEditingEnabled] = useState(false);

const [loading, setLoading] = useState(false);

const [userInfo, setUserInfo] = useState({

name: "",

phone\_number: "",

});

const handleUserInfoChange = ({ target: { name, value } }) => {

setUserInfo((prev) => ({ ...prev, [name]: value }));

};

const changeSkills = () => {

if (

addSkill !== "" &&

!skills.find((item) => item.toLowerCase() === addSkill.toLowerCase())

) {

setNewSkills((prev) => [...prev, addSkill.trim()]);

setSkills((prev) => [...prev, addSkill.trim()]);

}

setAddSkill("");

};const removeSkills = (skill\_name) => {

setRemovedSkills((prev) => [...prev, skill\_name]);

setSkills((prev) => prev.filter((item) => item !== skill\_name));

setNewSkills((prev) => prev.filter((item) => item !== skill\_name));

};

const updateSkills = async () => {

setLoading(true);

let skillsAdded = false,

skillsRemoved = false;

if (newSkills.length !== 0) {

skillsAdded = await saveUserSkills(newSkills, user.token);

}

if (removeSkills.length !== 0) {

skillsRemoved = await removeUserSkills(removedSkills, user.token);

}

if (skillsAdded || skillsRemoved) {

toast({

title: "Profile Updated!",

status: "info",

duration: 3000,

isClosable: true,variant: "left-accent",

position: "top",

});

}

setNewSkills([]);

setRemovedSkills([]);

setLoading(false);

};

const updateUserInfo = async () => {

setLoading(true);

const data = await updateUserDetails(userInfo, user.token);

if (data) {

setUser((prev) => {

prev = { ...prev, name: data.name, phone\_number: data.phone\_number };

localStorage.setItem("user", JSON.stringify(prev));

return prev;

});

toast({

title: "Profile Updated!",

status: "info",duration: 3000,

isClosable: true,

variant: "left-accent",

position: "top",

});

}

setLoading(false);

setIsEditingEnabled(false);

};

useEffect(() => {

if (user) {

(async () => {

setLoading(true);

let data = await getUserSkills(user?.token);

if (data) setSkills(data);

setLoading(false);

})();

setUserInfo({

name: user.name,

phone\_number: user.phone\_number,

});

}}, [user]);

return (

<>

{loading && <Progress size="xs" isIndeterminate colorScheme={"purple"}

/>}

<div className="my-5 mx-10">

<div className="border-2 border-blue-100 w-full h-fit rounded-xl p-5 flex

flex-col gap-3">

<div className="flex justify-between w-full min-h-[25vh]">

<div className="flex flex-col justify-between">

<h1 className="md:text-2xl text-xl font-medium flex items-center

gap-4">

Your Profile{" "}

<button>

{isEditingEnabled ? (

<AiOutlineClose

color="#ff8977"

onClick={() => setIsEditingEnabled(!isEditingEnabled)}

/>

) : (

<RiEdit2Fill

color="#4506cb"

onClick={() => setIsEditingEnabled(!isEditingEnabled)}

/>

)}

</button>

</h1>

<div className="flex flex-col gap-3">

{isEditingEnabled ? (primary"

primary"

primary"

<>

<input

name="name"

value={userInfo.name}

className="input input-bordered w-full input-xs p-3 text-lg input

type="text"

placeholder="name"

onChange={handleUserInfoChange}

/>

<input

disabled

value={user?.email}

className="input input-bordered w-full input-xs p-3 text-lg input

type="text"

placeholder="name"

/>

<input

name="phone\_number"

value={userInfo.phone\_number}

className="input input-bordered w-full input-xs p-3 text-lg input

type="number"

placeholder="phone number"

onChange={handleUserInfoChange}

/>

<button

className="btn btn-xs btn-outline btn-primary"

onClick={updateUserInfo}>

Update

</button>

</>

) : (

<>

<h2 className="md:text-2xl xl:text-2xl sm:text-xl">

{user?.name}

</h2>

<p className="md:text-xl sm:text-md text-gray-700">

{user?.email}

</p>

<span className="text-gray-700">{user?.phone\_number}</span>

</>

)}

</div>

</div>

<div className="flex flex-col justify-end w-fit gap-4">

<img

src="avatar.webp"

alt="profile"

className="md:w-36 w-20 rounded-md object-contain"

/>

</div>

</div>

<div className="divider my-2"></div>

<div className="flex flex-col">

<div className="flex justify-between gap-2 flex-col">

<h4 className="text-xl">Skills</h4><form

className="flex gap-5 items-center"

onSubmit={(e) => e.preventDefault()}

>

<input

autoComplete="off"

value={addSkill}

type="text"

name="addSkill"

placeholder="Add skills"

onChange={(e) => setAddSkill(e.target.value)}

className="input input-bordered w-full input-primary max-w-xl

input-sm"

/>

<button

className="hover:rotate-90 transition-all"

onClick={changeSkills}

>

<VscAdd size={20} />

</button>

</form>

{loading ? (

<Spinner

thickness="3px"

speed="0.65s"

emptyColor="gray.200"

color="blue.500"

size="md"className="m-3"

/>

) : (

<ul className="flex gap-2 flex-wrap">

{skills?.map((addSkill, ind) => (

<li

className="bg-indigo-100 rounded p-2 flex gap-2 items-center"

key={ind}

>

{addSkill}

<MdDeleteForever

color="#ff8977"

onClick={() => removeSkills(addSkill)}

size={20}

/>

</li>

))}

</ul>

)}

<button

className="btn btn-sm w-fit btn-primary"

type="button"

onClick={updateSkills}

>

Save

</button>

</div>

<div className="divider my-2"></div><div className="flex justify-between gap-2 flex-col">

<h4 className="text-xl">Resume/Portfolio</h4>

<div className="flex gap-5">

<input

className="input input-bordered w-full input-primary max-w-xl

input-sm"

type="text”

placeholder="paste the link"

/>

<button className="btn btn-primary btn-sm">update</button>

</div>

</div>

<div className="divider my-2"></div>

<div className="flex gap-2 flex-col">

<h3 className="text-xl">Socials</h3>

<div className="flex flex-col gap-2">

<div className="flex gap-5 items-center">

<GoMarkGithub size={20} />

<input

type="text"

placeholder="paste the link"

className="border-2 border-gray-300 rounded-md px-3 my-1 max-

w-md"

/>

</div>

<div className="flex gap-5 items-center">

<BsLinkedin size={20} />

<input

type="text"w-md"

w-md"

placeholder="paste the link"

className="border-2 border-gray-300 rounded-md px-3 my-1 max-

/>

</div>

<div className="flex gap-5 items-center">

<TfiTwitterAlt size={20} />

<input

type="text"

placeholder="paste the link"

className="border-2 border-gray-300 rounded-md px-3 my-1 max-

/>

</div>

<button className="btn btn-primary btn-sm max-w-fit">

save

</button>

</div>

</div>

</div>

</div>

</div>

</>

);

};

export default Profile;

**Dashboard.jsx**

import {

Progress,

SkeletonCircle,

SkeletonText,

Spinner,

} from "@chakra-ui/react";

import axios from "axios";

import React, { useContext, useEffect, useState } from "react";

import JobCard from "../components/JobCard";

import SearchBar from "../components/SearchBar";

import Skill from "../components/Skill";

import { AppContext } from "../context/AppContext";

import { getUserSkills} from "../proxies/backend\_api";

const Dashboard = () => {

const { user, skills, setSkills } = useContext(AppContext);

const [selectedSkills, setSelectedSkills] = useState([]);

const [skillsLoading, setSkillsLoading] = useState(false);

const [jobsLoading, setJobsLoading] = useState(false);

const [query, setquery] = useState("");

const [posts, setPosts] = useState(null);

const id = import.meta.env.VITE\_ADZUNA\_API\_ID;const key = import.meta.env.VITE\_ADZUNA\_API\_KEY;

const baseURL\_with\_skills =

`http://api.adzuna.com/v1/api/jobs/in/search/1?app\_id=${id}&app\_key=${key

}&results\_per\_page=15&what=${query}&what\_and=${selectedSkills.join(

" "

)}&&content-type=application/json`;

const baseURL =

`http://api.adzuna.com/v1/api/jobs/in/search/1?app\_id=${id}&app\_key=${key

}&results\_per\_page=15&what=${query}&content-type=application/json`;

const searchJobsFromQuery = async () => {

setJobsLoading(true);

if (query !== "" || !posts) {

const { data } = await axios.get(baseURL);

setPosts(data.results);

}

setJobsLoading(false);

};

const searchWithSkills = async () => {

setJobsLoading(true);

const { data } = await axios.get(baseURL\_with\_skills);

setPosts(data.results);setJobsLoading(false);

};

useEffect(() => {

if (user) {

(async () => {

setSkillsLoading(true);

setSkills(await getUserSkills(user.token));

setSkillsLoading(false);

})();

}

}, [user]);

useEffect(() => {

searchWithSkills();

}, [selectedSkills]);

useEffect(() => {

searchJobsFromQuery();

}, []);

return (

<>

{(jobsLoading || skillsLoading) && (

<Progress size="xs" isIndeterminate colorScheme={"purple"} />

)}

<div className="flex gap-10 m-10"><div className="hidden lg:flex bg-purple-600 w-1/5 p-5 h-3/6 rounded-lg

text-center flex-col gap-4">

<div className="text-2xl text-white capitalize font-extrabold">

Your skills

</div>

{skillsLoading ? (

<Spinner

className="self-center my-5"

thickness="3px"

speed="0.65s"

emptyColor="gray.200"

color="black.100"

size="lg"

/>

) : (

<ul className="list-none text-gray-200 flex flex-col gap-2">

{skills?.length === 0 ? (

<p className="text-gray-300">

Skills you add in the profile section will appear here!!

</p>

) : (

skills.map((skill, ind) => (

<Skill

skill={skill}

key={ind}

setSelectedSkills={setSelectedSkills}

disabled={skillsLoading}

/>

)))}

</ul>

)}

<p className="text-white text-sm">

(Include your skills in the search result)

</p>

</div>

<div className="mx-auto min-w-[80%] ">

<SearchBar setquery={setquery} onClick={searchJobsFromQuery} />

{query === "" ? (

<h2 className="text-2xl mt-5">Recommended Jobs</h2>

) : (

<h2 className="text-2xl mt-5">

Search for keywords {query}

{filterUsingSkills && `,${skills.join(",")}`}

</h2>

)}

5">

<div className="mt-5 grid grid-cols-1 lg:grid-cols-3 md:grid-cols-2 gap-

{jobsLoading

? [...new Array(10)].map((\_, ind) => (

<div key={ind}>

<SkeletonCircle size="8" className="mb-5" />

<SkeletonText

mt="4"

noOfLines={8}

spacing="4"color={"red"}

/>

</div>

))

: posts?.map((post, ind) => (

<JobCard

key={ind}

title={post.title}

company={post.company.display\_name}

description={post.description}

link={post.redirect\_url}

/>

))}

</div>

</div>

</div>

</>

);

};

export default Dashboard;

**Auth.jsx**

import { Tab, TabList, TabPanel, TabPanels, Tabs } from "@chakra-ui/react";

import React, { useContext, useEffect } from "react";

import { useNavigate } from "react-router-dom";

import Login from "../components/Login";

import SignUp from "../components/Signup";import { AppContext } from "../context/AppContext";

const Auth = () => {

const navigate = useNavigate();

const { user } = useContext(AppContext);

useEffect(() => {

if (user) navigate("dashboard");

}, []);

return (

<div className="flex flex-col justify-center items-center gap-10 mt-5">

<Tabs isFitted variant="line" colorScheme={"purple"}>

<TabList mb="1em">

<Tab>Login</Tab>

<Tab>SignUp</Tab>

</TabList>

<TabPanels>

<TabPanel>

<Login />

</TabPanel>

<TabPanel>

<SignUp />

</TabPanel>

</TabPanels>

</Tabs>

</div>

);};

export default Auth;

**helper.js**

export const emailRegex = /^[\w-.]+@([\w-]+\.)+[\w-]{2,4}$/;

export const urlRegex =

/((([A-Za-z]{3,9}:(?:\/\/)?)(?:[-;:&=\+\$,\w]+@)?[A-Za-z0-9.-]+(:[0-

9]+)?|(?:www.|[-;:&=\+\$,\w]+@)[A-Za-z0-9.-]+)((?:\/[\+~%\/.\w-\_]\*)?\??(?:[-

\+=&;%@.\w\_]\*)#?(?:[\w]\*))?)/;

export const BASE\_URL = import.meta.env.VITE\_BACKEND\_ENDPOINT;

**App.jsx**

import { useEffect } from "react";

import { HashRouter, Route, Routes } from "react-router-dom";

import Navbar from "./components/Navbar";

import { AppProvider } from "./context/AppContext";

import Auth from "./screens/Auth";

import Dashboard from "./screens/Dashboard";

import Profile from "./screens/Profile";

function App() {

useEffect(() => {

window.watsonAssistantChatOptions = {

integrationID: import.meta.env.VITE\_WATSON\_INTEGRATION\_ID, // The ID

of this integration.region: import.meta.env.VITE\_WATSON\_REGION, // The region your

integration is hosted in.

serviceInstanceID: import.meta.env.VITE\_WATSON\_SERVICE\_INSTANCE\_ID,

// The ID of your service instance.

onLoad: function (instance) {

instance.render();

},

};

setTimeout(function () {

const t = document.createElement("script");

t.src =

"https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +

(window.watsonAssistantChatOptions.clientVersion || "latest") +

"/WatsonAssistantChatEntry.js";

document.head.appendChild(t);

});

}, []);

return (

<HashRouter>

<AppProvider>

<Navbar />

<Routes>

<Route path="/" element={<Auth />} />

<Route path="/dashboard" element={<Dashboard />} />

<Route path="/profile" element={<Profile />} />

</Routes>

</AppProvider>

</HashRouter>

);}

export default App;

**main.jsx**

import { ChakraProvider } from "@chakra-ui/react";

import React from "react";

import ReactDOM from "react-dom/client";

import App from "./App";

import "./index.css";

ReactDOM.createRoot(document.getElementById("root")).render(

<React.StrictMode>

<ChakraProvider>

<App />

</ChakraProvider>

</React.StrictMode>

);

**Index.css**

@import

url("https://fonts.googleapis.com/css2?family=Ubuntu&display=swap");

@tailwind base;

@tailwind components;

@tailwind utilities;

:root {font-family: Inter, Avenir, Helvetica, Arial, sans-serif;

font-size: 16px;

line-height: 24px;

font-weight: 400;

color-scheme: light;

/\* color: rgba(255, 255, 255, 0.87);

background-color: #242424; \*/

font-synthesis: none;

text-rendering: optimizeLegibility;

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

-webkit-text-size-adjust: 100%;

}

\* {

margin: 0;

padding: 0;

font-family: "Ubuntu", sans-serif;

}

body::-webkit-scrollbar {

width: 5px;

background-color: none;

border-radius: 20px;

}

body::-webkit-scrollbar-thumb {background-color: #adadad;

border-radius: 20px;

}

body {

max-height: 100vh;

}

**Deployment.yaml**

## Enter your <docker\_username> before use

apiVersion: v1

kind: Service

metadata:

name: test

labels:

app: test

spec:

type: NodePort

ports:

- port: 5000

name: http

nodePort: 30080

selector:

app: app

---

apiVersion: extensions/v1beta1

kind: Deploymentmetadata:

name: test

spec:

replicas: 1

template:

metadata:

labels:

app: app

spec:

containers:

- name: ibm\_project

ports:

- containerPort: 5000

imagePullSecrets:

- name: regcred

**main.py**

from backend import create\_app

app = create\_app()

if name

== ' main ':

from waitress import serve

serve(app, port=5000)

**package.json**

{

"name": "react-flask-app",

"private": true,

"version": "0.0.0",

"type": "module",

"scripts": {

"start": "vite",

"build": "vite build",

"preview": "vite preview",

"server": "cd backend && flask --debug run"

},

"dependencies": {

"axios": "^1.1.3",

"daisyui": "^2.33.0",

"react": "^18.2.0",

"react-dom": "^18.2.0",

"react-icons": "^4.6.0",

"react-router-dom": "^6.4.2"

},

"devDependencies": {

"@types/react": "^18.0.17",

"@types/react-dom": "^18.0.6",

"@vitejs/plugin-react": "^2.1.0",

"autoprefixer": "^10.4.12",

"postcss": "^8.4.18",

"tailwindcss": "^3.1.8",

"vite": "^3.1.0"

} }

**postcss.config.cjs**

module.exports = {

plugins: {

tailwindcss: {},

autoprefixer: {},

},

}

**tailwind.config.cjs**

/\*\* @type {import('tailwindcss').Config} \*/

module.exports = {

darkMode: "class",

content: ["./index.html", "./src/\*\*/\*.{js,ts,jsx,tsx}"],

theme: {

extend: {},

},

plugins: [require("daisyui")],

daisyui: {

themes: ["light"],

},

};

**vite.config.js**

import react from "@vitejs/plugin-react";

import { defineConfig } from "vite";

// https://vitejs.dev/config/

export default defineConfig({

plugins: [react()],server: {

port: 3000,

cors: false,

},

});

**Dockerfile**

# Build step #1: build the React front end

FROM node:16-alpine as react-builder

WORKDIR /app

ENV PATH /app/node\_modules/.bin:$PATH

COPY package.json ./

COPY ./src ./src

COPY ./public ./public

COPY ./index.html ./vite.config.js ./postcss.config.cjs ./tailwind.config.cjs ./.env

./

RUN npm install

RUN npm run build

# Build step #2: build the API with the client as static files

FROM python:3.10

WORKDIR /app

COPY --from=react-builder /app/dist ./dist

COPY main.py ./main.py

RUN mkdir ./backend

COPY backend/ ./backend/

RUN pip install -r ./backend/requirements.txtEXPOSE 5000 s

ENTRYPOINT ["python","main.py"]

**GITHUB LINK:**

[IBM-EPBL](https://github.com/IBM-EPBL)/[**IBM-Project-40085-1660623206**](https://github.com/IBM-EPBL/IBM-Project-40085-1660623206)

**PROJECT DEMO LINK:**

**https://youtu.be/lyXvgz780\_A**