

	TABLE OF CONTENTS	
CHAPTER NO	TITLE	PAGE NO
1	INTRODUCTION	5
	1.1 PROJECT OVERVIEW	
	1.2 PURPOSE	
2	LITERATURE SURVEY	6
	2.1 EXISTING PROBLEM	
	2.2 REFERENCES	
	2.3 PROBLEM STATEMENT DEFINITION	
3	IDEATION & PROPOSED SOLUTION	8
	3.1 EMPATHY MAP CANVAS	
	3.2 BRAINSTORMING AND IDEA PRIORITIZATION	
	3.3 PROPOSED SOLUTION	
	3.3 PROBLEM STATEMENTS	
	3.4 PROBLEM SOLUTION FIT	
4	REQUIREMENTS ANALYSIS	12
	4.1 FUNCTIONAL REQUIREMENTS	
	4.2 NON-FUNCTIONAL REQUIREMENTS	
5	PROJECT DESIGN	14
	5.1 DATA FLOW DIAGRAMS	

	5.2 SOLUTION ARCHITECTURE	
	5.3 TECHNOLOGY ARCHITECTURE	
	5.4 USER STORIES	
6	PROJECT PLANNING & SCHEDULING	19
	6.1 SPRINT PLANNING & ESTIMATION	
	6.2 SPRINT DELIVERY SCHEDULE	
	6.3 REPORT FROM JIRA	
7	CODING & SOLUTIONING	24
	7.1 Feature 1	
	7.2 Feature 2	
	7.3 Feature 3	
	7.4 Database Schema	
8	TESTING	27
	8.1 Test Cases	
	8.2 User Acceptance Testing	
9	RESULTS	29
	9.1 Performance Metrics	
10	ADVANTAGES & DISADVANTAGES	30
11	CONCLUSION	32
12	FUTURE SCOPE	33
13	APPENDIX	34
	13.1 SOURCE CODE	

Cloud Application Development
Skill / Job Recommender Application

	13.2 GITHUB & PROJECT DEMO LINK	
--	---------------------------------	--

ABSTRACT

Recommendation system is a technique, which provides users with information, which he/she may have been interested in or accessed in the past. Traditional recommender techniques such as content and collaborative filtering used in various applications such as education, social media, marketing, entertainment, e-governance and many more. Content-based and collaborative filtering has many advantages and disadvantages and they are useful in specific applications. Sparsity and cold start problems are major challenges in content and collaborative filtering. Challenges of content and collaborative filtering can be solved by using hybrid filtering. In our project, we use Hybrid filtering which combines the features of two recommender systems like content and collaborative; content-based filtering improves the classification accuracy and collaborative model easily gives the best-predicted result of a latent factor model. The combination of the two techniques is used to achieve better job and skill recommendations.

CHAPTER - 5

INTRODUCTION

INTRODUCTION

Recommendation system is a technique, which provides users with information, which he/she may have been interested in or accessed in the past. Traditional recommender techniques such as content and collaborative filtering used in various applications such as education, social media, marketing, entertainment, e-governance and many more. Content-based and collaborative filtering has many advantages and disadvantages and they are useful in specific applications. Sparsity and cold start problems are major challenges in content and collaborative filtering. Challenges of content and collaborative filtering can be solved by using hybrid filtering. Hybrid filtering combines the features of two recommender systems like content and collaborative; content-based filtering improves the classification accuracy and collaborative model easily gives the best-predicted result of a latent factor model.

CHAPTER - 2

LITERATURE SURVEY

2.1 EXISTING PROBLEM

Gathering the details from the user's profile, CV's and creating job recommendations and matching the profile for the candidate recommendations. The candidates profile details will match with the candidate recruitment profile for providing recommendation matching information. According to the professional recommender system the profiles will be consolidated for getting the confidential information. Recommended job and candidate information can be changed according to the user's recruitment needs and user's updated profile details. Recommendation details available in search mode also, jobs recommendations can be search area wise, city wise and state wise, domain interest.

2.2 REFERENCES

1. [Shaha T Al-Otaibi and Mourad Ykhlef. 2012. A survey of job recommender systems. International Journal of Physical Sciences , Vol. 7, 29 \(2012\), 5127--5142](#)
2. [Technical Job Recommendation System Using APIs and Web Crawling](#)
3. [Skill Scanner: Connecting and Supporting Employers, Job Seekers and Educational Institutions with an AI-based Recommendation System by Koen Bothmer and Tim Schlippe](#)
4. [A Life-long Learning Recommender System to Promote Employability](#)

2.3 PROBLEM STATEMENT DEFINITION

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	Unemployed	Find a job	It takes a long time	I cannot find the right one	Demotivated
PS-2	Unemployed	Find a job	I cannot find a job at my desired location	I am searching for jobs offline	Helpless
PS-3	Unemployed	Find a job	I cannot find a job with expected CTC	I do not have enough contacts with big companies	Disappointed
PS-4	Unemployed	Find a job	It takes a long time	I am a disabled person	Hopeless

CHAPTER -3

IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

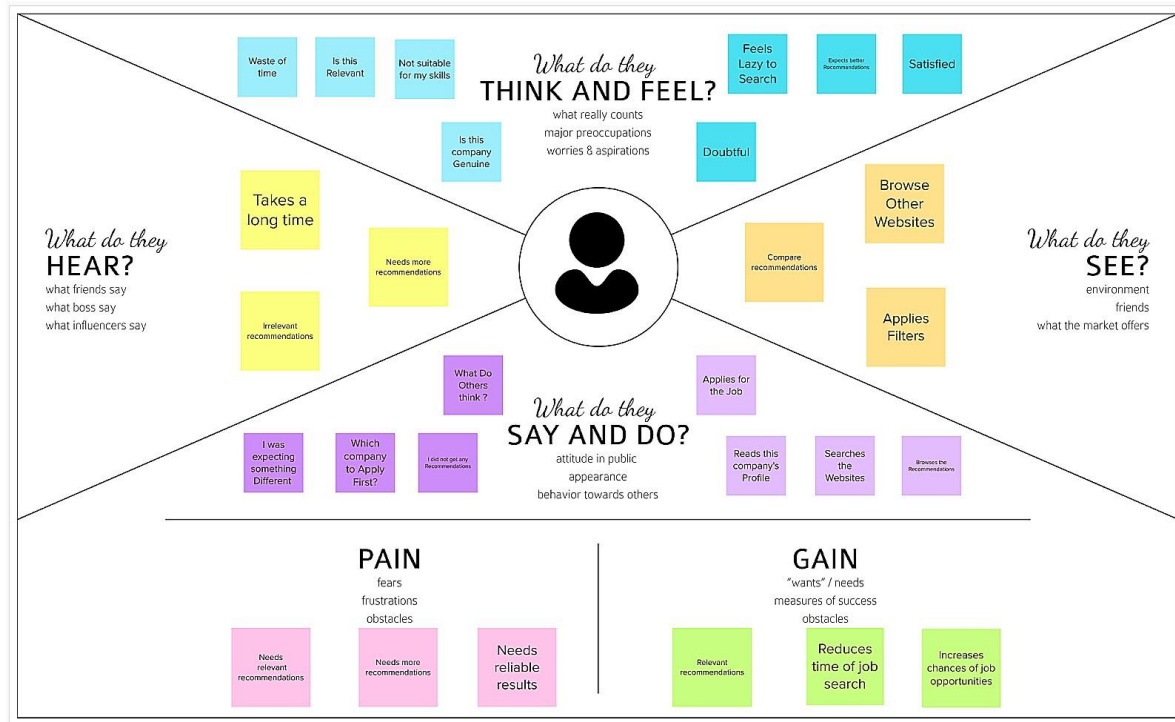


Fig 3.1 Empathy Map canvas

3.2 BRAINSTORMING AND IDEA PRIORITIZATION



Fig 3.2 Brainstorming And Idea Prioritization

3.3 PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Jobs must be recommended based on the skills of the candidate
2.	Idea / Solution description	A Job recommender using Hybrid filtering technique
3.	Novelty / Uniqueness	Use of Hybrid filtering which is a combination of both collaborative and content based filtering
4.	Social Impact / Customer Satisfaction	Getting relevant and more recommendations which can improve chances of employment
5.	Business Model (Revenue Model)	Ads for companies who pay for the web application and certain revenue for reaching a certain number of registrations.
6.	Scalability of the Solution	Kubernetes allows users to scale the total containers used based on application requirements while conventional servers cannot be scalable.

3.4 PROBLEM STATEMENTS

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	Unemployed	Find a job	It takes a long time	I cannot find the right one	Demotivated
PS-2	Unemployed	Find a job	I cannot find a job at my desired location	I am searching for jobs offline	Helpless
PS-3	Unemployed	Find a job	I cannot find a job with expected CTC	I do not have enough contacts with big companies	Disappointed
PS-4	Unemployed	Find a job	It takes a long time	I am a disabled person.	Hopeless

3.5 PROBLEM SOLUTION FIT

Project Title: Skill / Job Recommender Application Project Design Phase-I - Solution Fit Team ID: PNT2022TMID30155

Define CS, fit into CL	1. CUSTOMER SEGMENT(S) CS 1. Job Seekers (Experienced) 2. Freshers	6. CUSTOMER LIMITATIONS CL <small>EG. BUDGET, DEVICES</small> 1. Network Connection 2. Customer's desired company must have online hiring	5. AVAILABLE SOLUTIONS AS <small>PROS & CONS</small> 1. Use of Hybrid filtering which uses both Content Based and Collaborative Filtering techniques to overcome the difficulties of both techniques.	Explore AS, differentiate
	2. PROBLEMS / PAINS + ITS FREQUENCY PR 1. Irrelevant recommendations 2. Sparse recommendations	9. PROBLEM ROOT / CAUSE RC 1. Cold start problem of Collaborative Filtering 2. Not enough views on a particular profile	7. BEHAVIOR + ITS INTENSITY BE 1. Tries another job recommending website	
Identify strong TR & EM	3. TRIGGERS TO ACT TR 1. Hearing about the website through friends, recruiters or social media	10. YOUR SOLUTION SL To generate relevant and more recommendations according to the user's needs.	8. CHANNELS of BEHAVIOR CH ONLINE Other Job recommending websites such as LinkedIn, Naukri etc.,	Extract online & offline CH of BE
	4. EMOTIONS <small>BEFORE / AFTER</small> EM Before: Frustrated, Hopeless After: Hopeful, Confident		OFFLINE Asks friends or colleagues for references to attend interview at desired company	

Fig 3.5 Problem Solution Fit

CHAPTER-4

REQUIREMENTS ANALYSIS

4.1 Functional Requirements:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIn
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Login	Login using credentials
FR-4	User Search	Search for desired company
FR-5	User Profile	Complete user profile by providing personal details
FR-6	User Application	User applies for the desired company

4.2 Non-functional Requirements:

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Filters for the acquired results
NFR-2	Security	Two step verification
NFR-3	Reliability	Applicants can access their resume 98% of the time without failure

Cloud Application Development
Skill / Job Recommender Application

NFR-4	Performance	The website's loading time should be less than 5 seconds
NFR-5	Availability	Companies can post jobs on the website throughout the week at any time during the day
NFR-6	Scalability	The solution shall be able to support an annual growth of 10% of new customers.

CHAPTER-5

PROJECT DESIGN

5.1 DATA FLOW DIAGRAM

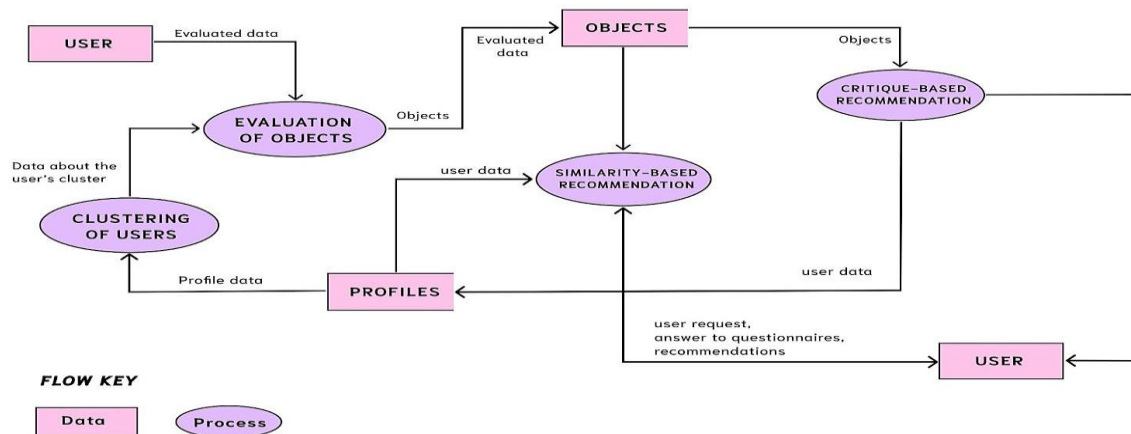


Fig 5.1 Data Flow Diagram

5.2 SOLUTION ARCHITECTURE

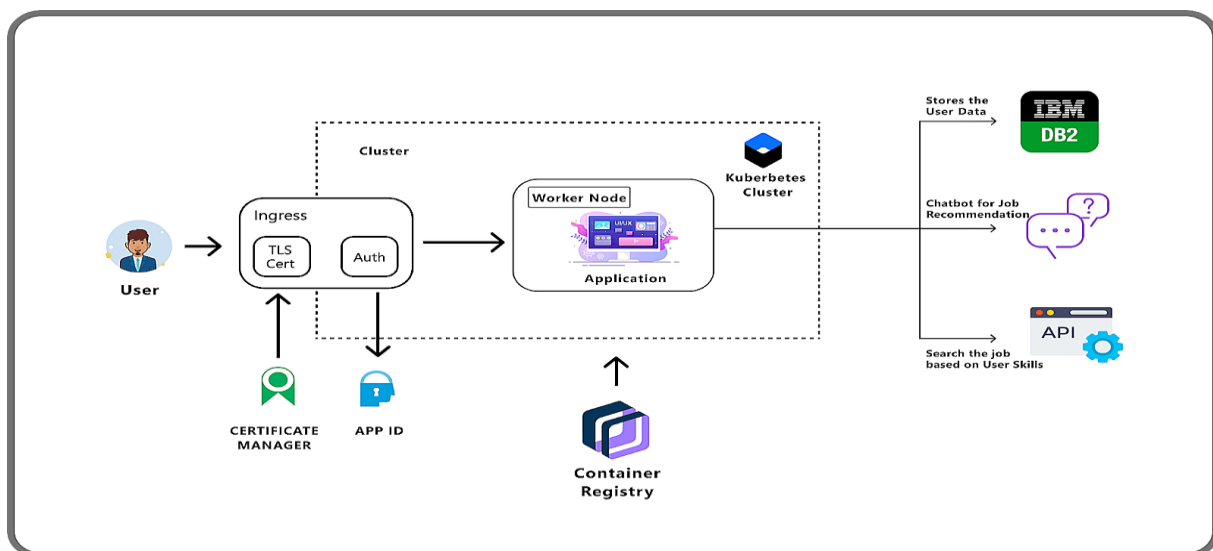


Fig 5.2 Solution Architecture

5.3 TECHNOLOGY ARCHITECTURE

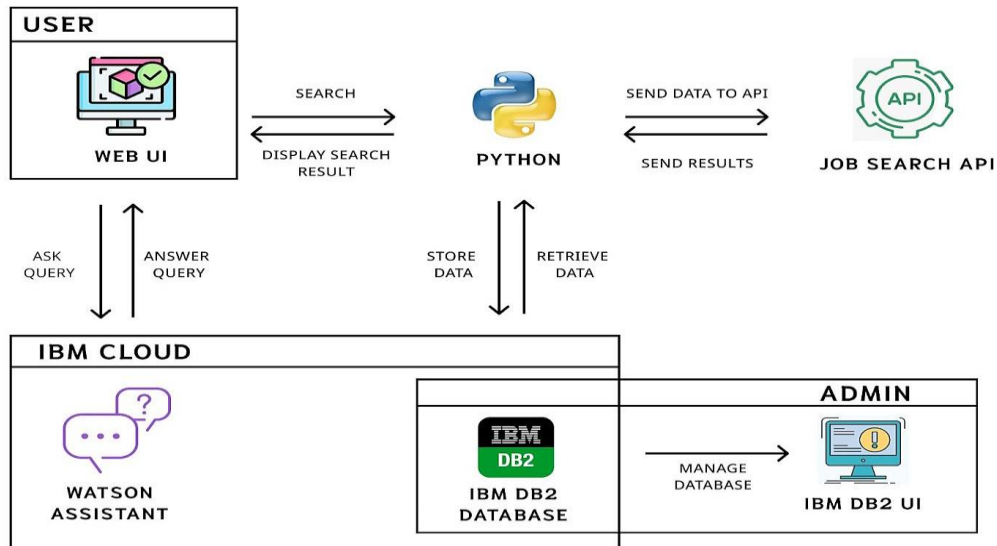


Fig 5.3 Technology Architecture

5.4 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web user)	Registration	USN-1	As a user, I can register for an account by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2

		USN-4	As a user, I can register for the application through Gmail	I can register & access the dashboard with Gmail Login	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can access the dashboard	High	Sprint-1
	Search	USN-6	As a user, I can search for the desired companies	Companies related to the search terms are listed	High	Sprint-2
	Apply	USN-7	As a user, I can apply for a company	Application is submitted to the company	High	Sprint-2
	Review	USN-8	As a user, I can review the company	Review is listed on the company's profile	Medium	Sprint-2

Cloud Application Development
Skill / Job Recommender Application

Admin	Forward	USN-9	As an admin, I must forward the applications to the respective companies	The application is received by the company	High	Sprint-1
	Send Confirmation	USN-10	Confirmation mail is sent from the respected company	Confirmation is received by the user	High	Sprint-2
	Manage Review	USN-11	As an admin, I must make the reviews appear on the company's profile	Reviews appear on the company's page	Low	Sprint-2

CHAPTER-6

PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	5	High	Lakshman S, Nandha Kumar BK
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	5	High	Nandha kumar B K, Mugesh G
Sprint-2		USN-3	As a user, I can register for the application through Facebook	3	Low	Karthik Raja R
Sprint-2		USN-4	As a user, I can register for the application through Gmail	5	Medium	Lakshman S
Sprint-1	Login	USN-5	As a user, I can log into the application by entering email & password	5	High	Mugesh G, Lakshman S, Karthik Raja R

Cloud Application Development
Skill / Job Recommender Application

Sprint-2	Search	USN-6	As a user, I can search for the desired companies	7	High	Nandha Kumar B K , Mugesh G
Sprint-3	Apply	USN-7	As a user, I can apply for a company	6	High	Lakshman S, Karthik Raja R
Sprint-3	Review	USN-8	As a user, I can review the company	4	Medium	Nandha Kumar B K
Sprint-4	Forward	USN-9	As an admin, I must forward the applications to the respective companies	4	High	Mugesh G, Lakshman S
Sprint-4	Send Confirmation	USN-10	Confirmation mail is sent from the respected company	4	High	Nandha Kumar BK, Karthik Raja R
Sprint-4	Manage Review	USN-11	As an admin, I must make the reviews appear on the company's profile	1	Low	Lakshman S
Sprint-4	Chatbot	USN-12	As a user, I can interact with Watson Assistant to resolve my queries	1	Low	Nandha Kumar B K

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	15	6 Days	24 Oct 2022	29 Oct 2022	15	29 Oct 2022
Sprint-2	15	6 Days	31 Oct 2022	05 Nov 2022	15	31 Oct 2022
Sprint-3	10	6 Days	07 Nov 2022	12 Nov 2022	15	07 Nov 2022
Sprint-4	10	6 Days	14 Nov 2022	19 Nov 2022	10	14 Nov 2022

VELOCITY:

Sprint-1 and Sprint-2

$$AV = \text{sprint duration} / \text{velocity} = 156 = 2.5$$

Sprint-3 and Sprint-4

$$AV = \text{sprint duration} / \text{velocity} = 106 = 1.66$$

6.3 REPORT FROM JIRA

6.3.1 Sprint 1:

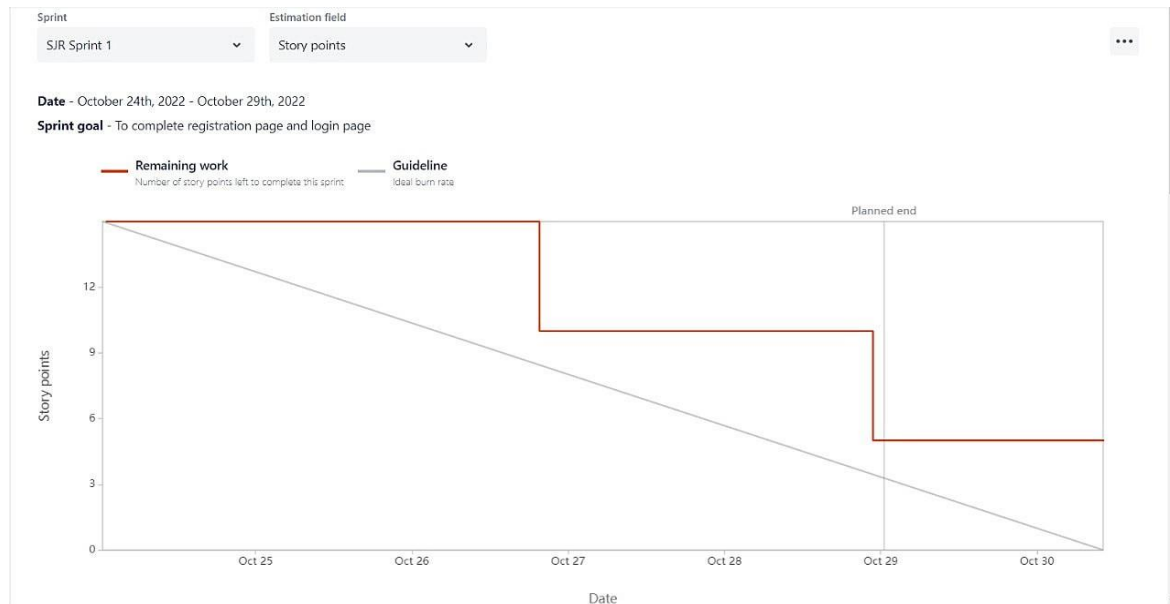


Fig 6.3.1 Sprint 1

6.3.2 Sprint 2:

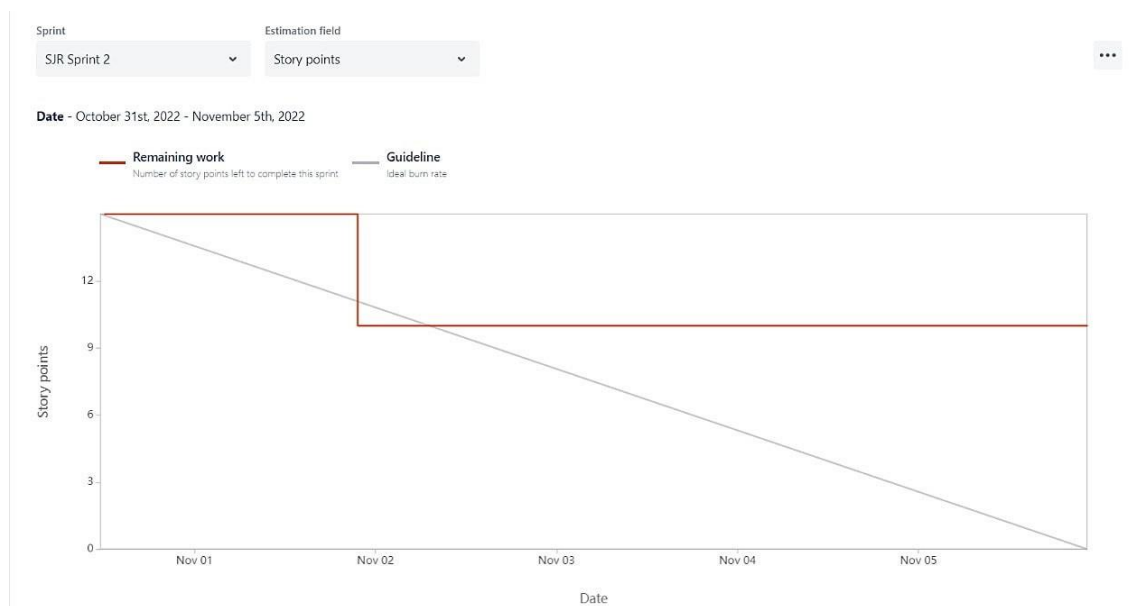


Fig 6.3.2 Sprint 2

6.3.3 Sprint 3:

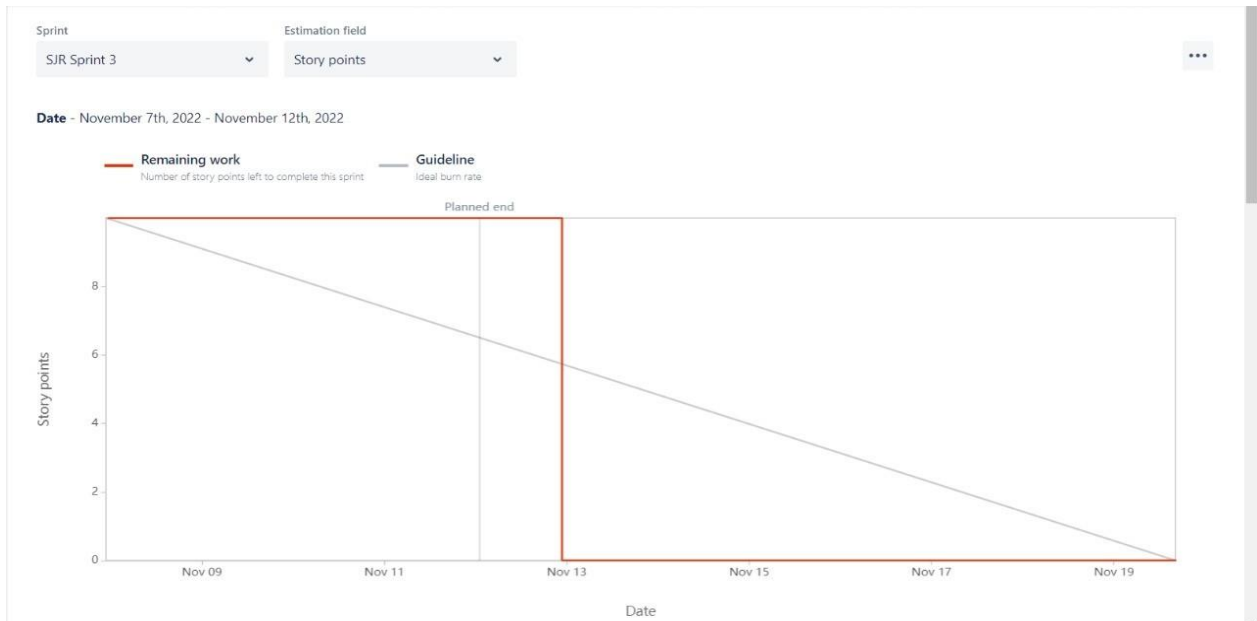


Fig 6.3.3 Sprint 3

6.3.4 Sprint 4:

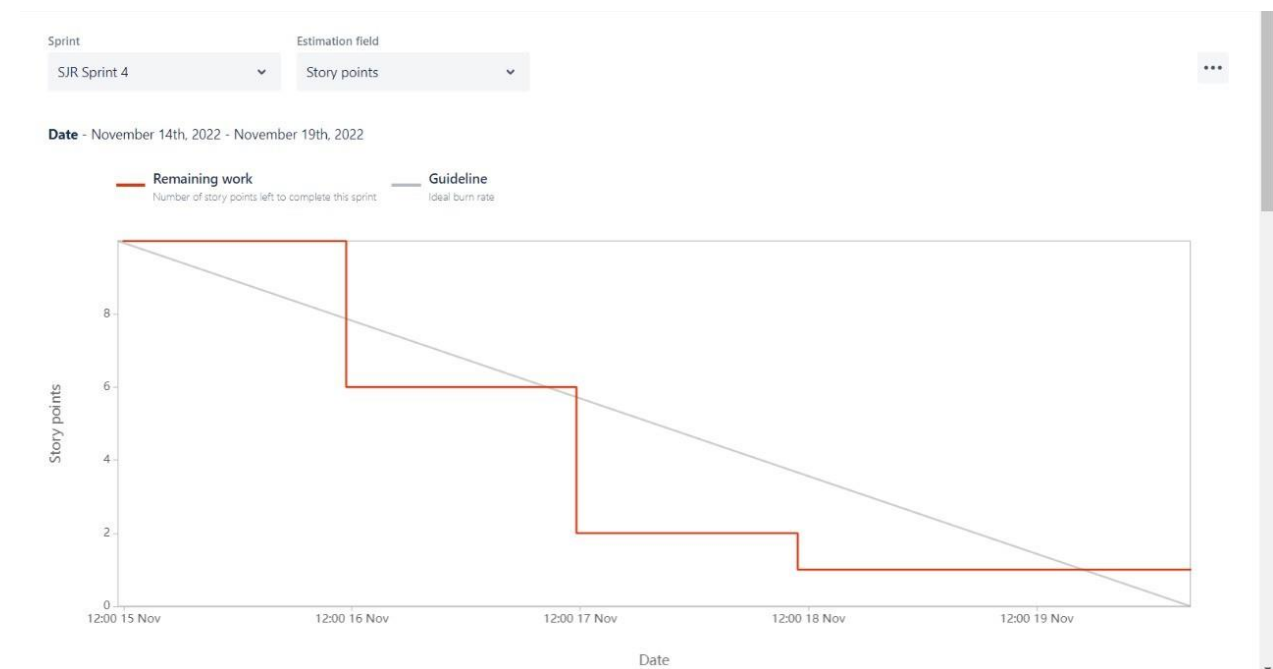


Fig 6.3.4 Sprint 4

CHAPTER - 7

CODING & SOLUTIONING

7.1 FEATURE 1

REGISTRATION PAGE

- The registration page consists of details to be filled by the user such as first name, last name, email ID, password and confirm password.
- If the entered details are valid and do not exist already, then the user is redirected to the verification page where he/she is required to enter the OTP sent to their email ID.
- On successful validation, the user is redirected to the login page where he can login with his credentials.
- If the OTP was not received, then OTP can be resent with the “Resend verification mail” link.

LOGIN PAGE

- The login page consists of email ID and password fields which are used to verify the identity of the user
- The login page consists of “SignIn with Google” feature which can be used to login to the web application
- It also consists of “Forgot Password” feature which helps the user to reset their password if they forget it.

PROFILE PAGE

- On successful login, the user is redirected to the profile page where he is expected

to provide the details required by the companies and click the “Save Profile” button to finish his profile.

- The profile page consists of details such as first name, last name, email ID, educational qualifications, mobile number, experience and preferred skill.

7.2 FEATURE 2

SEARCH FEATURE

- After completion of the profile, the user is redirected to the home page where he can search for companies based on job title and skill.
- Jobs are recommended to the user according to the skill which they have provided in their profile
- After finding the job which they like, they can click the “Apply Job” button to apply for the job

APPLY JOB PAGE

- The apply job page consists of the user’s personal information and educational qualifications and he clicks the “Apply Job” button to apply for the job
- A confirmation mail is sent to the user’s registered email id

7.3 FEATURE 3

ADMIN PAGE

- Like the user login, the web application also consists of admin login which can be used by the admin to post a new job
- After a job is posted, all the users possessing the specific skill for the job are notified through their registered email ID

CHATBOT

- The user can interact with the chatbot to find the companies which are hiring for the specific skill which he possesses.

7.4 DATABASE SCHEMA

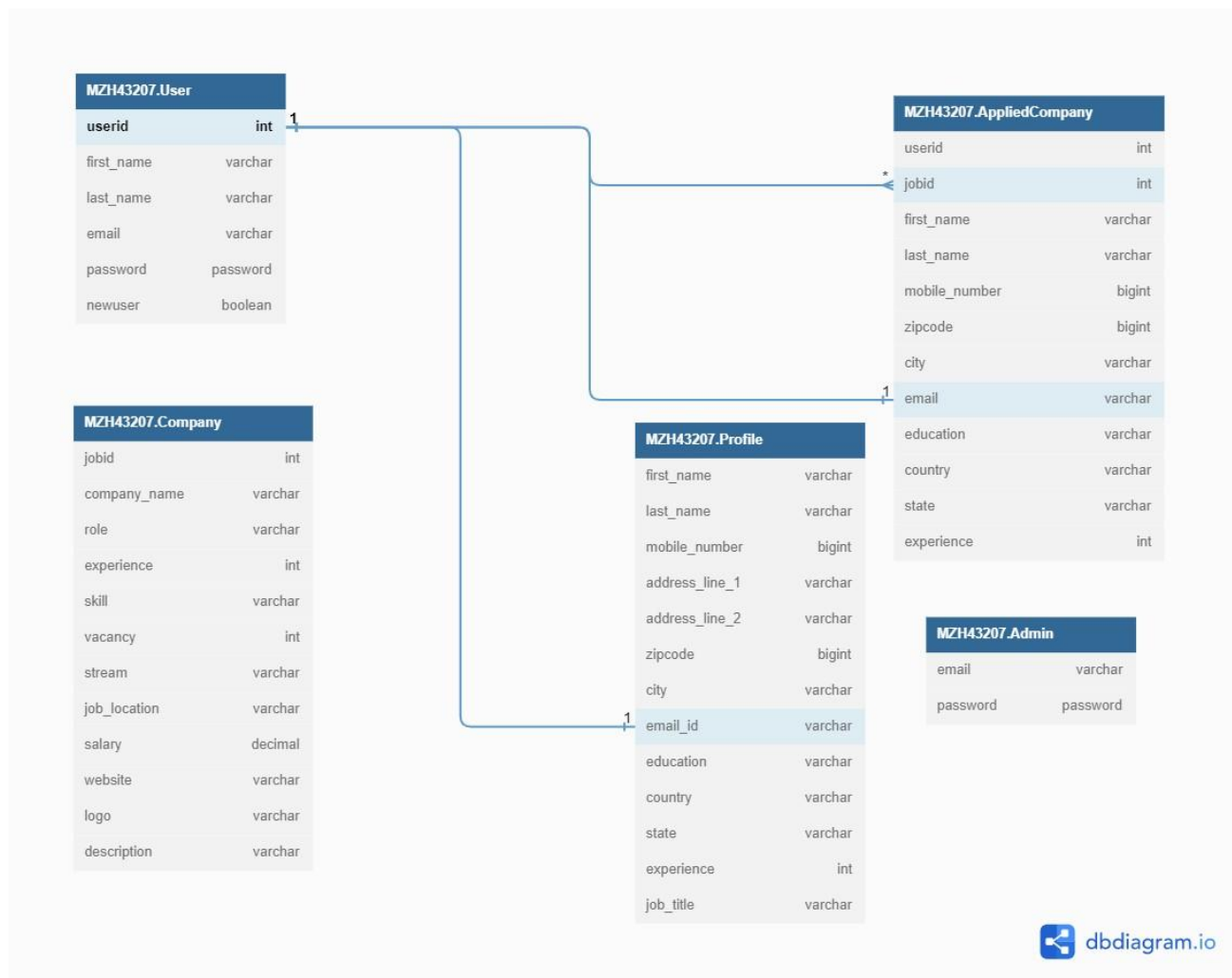


Fig 7.3 Database Schema

CHAPTER - 8

TESTING

8.1 TEST CASES

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID
Registration Page_TC_003	Functional	Registration Page	Check if password and confirm password are same		1. Enter different passwords for Password and Confirm Password fields	password - kanth@123 confirm password - kanth@123	It should display a message saying that the passwords don't match	Working as expected	Pass	No Error	N	
Registration Page_TC_004	Functional	Registration Page	Check if the email is valid		1. Enter Invalid Emails 2. Click on the Register Button.	email - lakshmikanth@xyz.com	It should show an invalid email message	Not Working as expected	Fail	It is not possible to collect all the domain name existing in this world for a personal scale project	N	BUG_ID-001
Registration Page_TC_005	Functional	Registration Page	Check if Email already exists in the database.	User must be already register	1. Enter an already registered email. 2. Click Register button	email - lakshmikanth@gmail.com password - kanth@123	It should say that email already exists	Working as expected	Pass	No Error	N	
Login Page_TC_006	UI	Login Page	1. Click textboxes, checkboxes and buttons		1. Click textboxes, checkboxes and buttons		UI should work properly	Working as expected	Pass	No Error	N	
Login Page_TC_007	Functional	Login Page	Check user should be filling all the required fields		1. Enter valid values in the required fields. 2. Click the Login button.	email - lakshmikanth@gmail.com password - kanth@123	1. Users should be logged in successfully 2. User should be redirected to home page	Working as expected	Pass	No Error	N	
Login Page_TC_008	Functional	Login Page	Check if email and passwords is invalid	User must be register	1. Enter Invalid email 2. Enter Invalid password 3. Click on the Login button	email - lakshmikanth@xyz.com password - kanth123	It should show invalid email or invalid password message	Working as expected	Pass	No Error	N	
Profile Page_TC_009	UI	Profile Page	Check all textboxes, checkboxes and buttons		1. Click textboxes, checkboxes and buttons		UI should work properly	Working as expected	Pass	No Error	N	
Profile Page_TC_010	Functional	Profile Page	User should fill all the required fields	User must be logged in	1. Enter valid values in the required fields. 2. Click the Save button.	Lakshmikanth, R, 1234567890, Four Roads, 636004, Salem, B.E CSE, India, Tamil Nadu, 3, Python	1. Users successfully save his profile 2. User should be redirected to home page	Working as expected	Pass	No Error	N	

Profile Page_TC_011	Functional	Profile Page	Users can view his profile		1. Click on the profile button		It should show an user profile with details which he entered previously	Working as expected	Pass	No Error	N	
Profile Page_TC_012	Functional	Profile Page	Users can update his profile		1. Click on the profile button 2. Enter valid values in the required fields. 3. Click the Save button.		1. Users successfully updated his profile 2. User should be redirected to home page	Working as expected	Pass	No Error	N	
Admin Page_TC_013	UI	Admin Page	Check all textboxes, checkboxes and buttons		1. Click textboxes, checkboxes and buttons		It should say that email already exists	Working as expected	Pass	No Error	N	
Admin Page_TC_014	Functional	Admin Page	Admin should fill all the required fields		1. Enter valid values in the required fields. 2. Click the Login button.	email - hiremeadmin@gmail.com password - hire@123	1. Users should be logged in successfully 2. User should be redirected to Admin home page	Working as expected	Pass	No Error	N	
Admin Page_TC_015	Functional	Admin Page	Check if Email and Password is invalid		1. Enter Invalid Email and password 2. Click on the Login button	email - hireme@gmail.com password - hire123	It should show invalid email or password message	Working as expected	Pass	No Error	N	
Admin Page_TC_016	Functional	Admin Page	Admin can post new jobs	Admin must be logged in	1. Enter valid values in the required fields. 2. Click the Post button.	Zoho Corporation, Software Developer, 0, Java, C++, Python, 20, Any Stream, Chennai, Bangalore, 7, https://www.zoho.com/, https://www.zoho.com/branding/images/zoho-logo.png	1. It is visible to the users, and they can apply it 2. It will send a suggestion email to the users based on their skill	Working as expected	Pass	No Error	N	
Apply Page_TC_017	UI	Apply Page	Check all textboxes, checkboxes and buttons		1. Click textboxes, checkboxes and buttons		UI should work properly	Working as expected	Pass	No Error	N	

Apply Page_TC_018	Functional	Apply Page	User should fill all the required fields		1. Enter valid values in the required fields. 2. Click the Apply button.		1. User successfully applied for a job and receive a confirmation email. 2. User should be redirected to home page	Working as expected	Pass	No Error	N	
Apply Page_TC_019	Functional	Apply Page	It will fill the user details automatically based on the user profile	must not have applied before	1. Select the company and click the apply for a job button.		User details are filled automatically and phone number can be updated	Working as expected	Pass	No Error	N	
Apply Page_TC_020	Functional	Apply Page	The user does not permit applying for a job with the same company		1. The user should choose the company which he has previously applied 2. Click the Apply button		It will display that the user has already applied for this job	Not Working as expected	Fail	Internal JavaScript error	N	BUG_ID-002
Home Page_TC_021	UI	Home Page	Check all textboxes, checkboxes and buttons		1. Click textboxes, checkboxes and buttons		UI should work properly	Working as expected	Pass	No Error	N	
Home Page_TC_022	Functional	Home Page	Jobs will be displayed on the home page based on user-provided details.		1. Users should complete the profile. 2. Click the Save button.		1. A list of available jobs will display on the home page 2. The user can view more information about the company by selecting it	Working as expected	Pass	No Error	N	
Home Page_TC_023	Functional	Home Page	User should fill the search fields		1. Enter values in the required fields. 2. Click the Search button.	Java	1. It will display a list of jobs based on the user's search 2. The user can view more information about the company by selecting it	Working as expected	Pass	No Error	N	

Home Page_TC_024	Functional	Home Page	Suggest to the user while he is looking for		1. Enter a search query 2. Click the submit button		1. Suggestions show up while typing. 2. Search results are displayed after hitting the submit button. 3. No results are displayed if there is no such term	Working as expected	Pass	No Error	N	
------------------	------------	-----------	---	--	---	--	--	---------------------	------	----------	---	--

8.2 USER ACCEPTANCE TESTING:

Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the [ProductName] project at the time of the release to User Acceptance Testing (UAT).

Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved.

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	2	4	1	3	10
Duplicate	1	1	0	0	2
External	1	0	0	0	1
Fixed	1	1	2	5	9
Not Reproduced	0	0	0	0	0
Skipped	0	0	1	1	2
Won't Fix	0	0	2	1	3
Totals	5	6	6	10	25

Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Registration Page	7	0	0	7
Login Page	5	0	0	5
Profile Page	5	0	0	5
Admin Page	6	0	0	6
Apply Page	5	0	0	5
Home Page	5	0	0	5

CHAPTER - RESULTS

PERFORMANCE METRICS:

				NFT - Risk Assessment					
S.No	Project Name	Scope/feature	Functional Changes	Hardware Changes	Software Changes	Impact of Downtime	Load/Volumen Changes	Risk Score	Justification
1	Skill and Job Recommender	New	Moderate	No Changes	Moderate		>30 to 50 %	ORANGE	changes have been absorbed
2	Skill and Job Recommender	New	High	No Changes	Moderate		>50 to 70%	RED	changes have been absorbed
3	Skill and Job Recommender	New	Low	No Changes	Moderate		>10 to 30%	GREEN	changes have been absorbed
			NFT - Detailed Test Plan						
			S.No	Project Overview	NFT Test approach	Options/Dependencies	Approvals/SignOff		
			1 Skill and Job Recommender	Performance	Low				
			End Of Test Report						
S.No	Project Overview	FT Test approach	NFR - Met	Test Outcome	GO/NO-GO decision	Recommendations	Identified Defects (Detected/Closed/Open)	Approvals/SignOff	
1	Skill and Job Recommender	Performance	yes	Good		Reduce calls to Database	closed		

CHAPTER - 10

ADVANTAGES & DISADVANTAGES

ADVANTAGES:

Getting Job Alerts:

A superior job portal provides standard job alerts whenever there are job openings for you. This way, you won't neglect a chance to be valid for your target job and bound to start your career. Also, you can discover more posts to increase better options.

Classified:

Once you register in a job portal, all your achievements, skills, and individual details will be reserved privately. This will be kept secure unless you let them split your details to your future employers. Also, job portals keep your job search narration classified which permits you to do the task surreptitiously.

More job opportunities:

Job portals offer a full collection of job choices from top companies. It means you have a superior chance of searching the job you want anywhere and however; you want it.

Resourceful:

With these portals, you can be relevant for a job effortlessly. Instead of going to your intention firm, you can submit your resume online. After submitting, you can relax, and wait for your future employers to take action.

DISADVANTAGES:

Competition:

A noteworthy weakness to job chasing on the web is that every other person is doing it. You're one of the a great many individuals competing for jobs and maybe thousands applying for a similar job on the grounds that the accommodation factor bids to the majority of your rivals also. Even when you're hunting down jobs outside your region, you're rivaling job-searchers in that area and individuals applying for a similar job from each other locale.

Building an audience:

Building an audience for an online job recommendation website is quite hard because there will not be many people who know about our website in the beginning. Building an audience happens naturally when people recommend our website their friends, colleagues and fellow jobseekers which can happen immediately or take a long time. It is a factor which we cannot control.

CHAPTER - 10

CONCLUSION

Job Search Portals stands as a revolutionizing element in the sphere of recruitment. They act as a communication bridge between applicants and recruiters facilitating their requirements. This application helps organizations to have a greater exposure to the candidate pool and also job seekers facilitating wide search of jobs matching their interests. It provides user friendly interface which facilitates in reaching wide range of audience. The application has achieved all the requirements that were initially set in the requirements gathering phase. Starting from requirements elicitation to design, construction, implementation and testing, I have gained a very good experience working with various technologies at every phase. Development of this project boosted my confidence in web development.

CHAPTER - 10

FUTURE SCOPE

This project fulfills the primary requirements of the job seekers and employers. It can be extended in several ways – the job seekers might be interested in building a strong Resume, we can provide tips and information for the same. We can also provide templates for building the Resumes which might interest most applicants. We can also extend the website by providing review feature which helps the job seekers know more about a company's recruitment process.

CHAPTER - 13

APPENDIX

13.1 SOURCE CODE

app.py :

```
File Edit Selection View Go Run Terminal Help app.py - Project Development Phase - Visual Studio Code
app.py
Sprint 3 > app.py
1 import csv
2 import json
3 import os
4 import pathlib
5 from random import randint
6
7 import google.auth.transport.requests
8 import ibm_db
9 import requests
10 from flask import Flask, abort, redirect, render_template, request, session, url_for
11 from flask_mail import Mail, Message
12 from google.oauth2 import id_token
13 from google_auth_oauthlib.flow import Flow
14 from pip_vendor import cachecontrol
15
16 connectionstring = "DATABASE=bludb;HOSTNAME=21fecfd8-47b7-4937-840d-d791d0218660.bs21o90188kqb1od81cg.databases.appdomain.cloud;PORT=31864;PROTOCOL=TCPIP;UID=mzh43207;PWD=PLYMGF"
17
18 connection = ibm_db.connect(connectionstring, '', '')
19 app = Flask(__name__)
20 app.debug = True
21
22 mail = Mail(app)
23 app.secret_key = "HireMe.com"
24
25 first_name = ""
26 last_name = ""
27 password = ""
28
29
30
31 app.config["MAIL_SERVER"] = 'smtp.gmail.com'
32 app.config["MAIL_PORT"] = 465
33 app.config["MAIL_USERNAME"] = '2k19cse052@kilot.ac.in'
34 app.config["MAIL_PASSWORD"] = ''
35 app.config["MAIL_USE_TLS"] = False
36 app.config["MAIL_USE_SSL"] = True
37 mail = Mail(app)
38
39 os.environ["OAUTHLIB_INSECURE_TRANSPORT"] = "1"
40
```

```
File Edit Selection View Go Run Terminal Help app.py - Project Development Phase - Visual Studio Code
app.py
Sprint 3 > app.py
41
42 GOOGLE_CLIENT_ID = ""
43 client_secrets_file = os.path.join(
44     pathlib.Path(__file__).parent, "client_secret.json")
45
46 flow = Flow.from_client_secrets_file(
47     client_secrets_file=client_secrets_file,
48     scopes=["https://www.googleapis.com/auth/userinfo.profile",
49             "https://www.googleapis.com/auth/userinfo.email", "openid"],
50     redirect_uri="http://127.0.0.1:5000/callback"
51 )
52
53
54 @app.route("/signup")
55 @app.route("/")
56 def signup():
57     return render_template("signup.html")
58
59
60 @app.route('/verification', methods=["POST", "GET"])
61 def verify():
62     global first_name
63     global last_name
64     global password
65     global otp
66
67     if request.method == 'POST':
68         first_name = request.form.get('first_name')
69         last_name = request.form.get('last_name')
70         password = request.form.get('password')
71         useremail = request.form.get('email')
72         sql = "SELECT * FROM User WHERE email =?"
73         stmt = ibm_db.prepare(connection, sql)
74         ibm_db.bind_param(stmt, 1, useremail)
75         ibm_db.execute(stmt)
76         account = ibm_db.fetch_assoc(stmt)
77
78         if (account):
79             return render_template('signup.html', msg="You are already a member, please login using your details")
80
```

Cloud Application Development Skill / Job Recommender Application

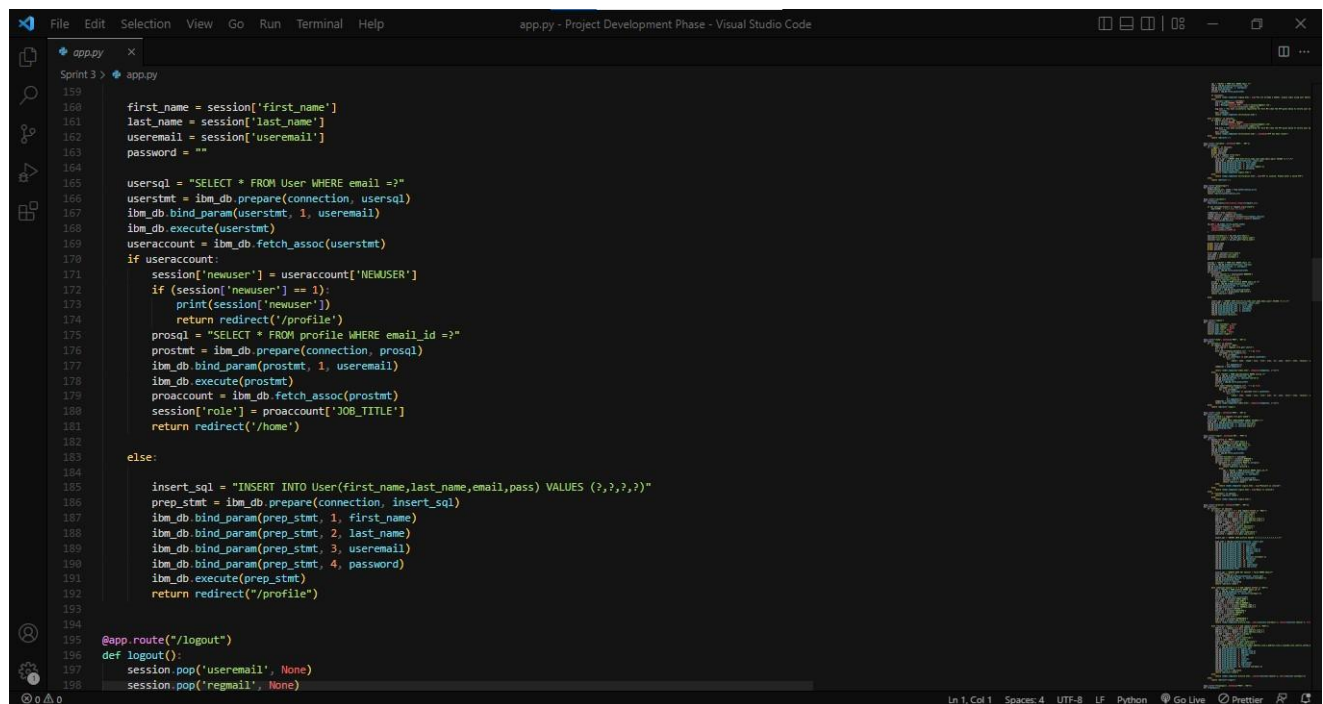
```
File Edit Selection View Go Run Terminal Help app.py - Project Development Phase - Visual Studio Code

app.py
Sprint 3 > app.py
80     else:
81         session['regmail'] = useremail
82         otp = randint(900000, 999999)
83         msg = Message(subject='OTP', sender='hackjacks@gmail.com',
84                       recipients=[session['regmail']])
85         msg.body = "You have succesfully registered for Hire Me!\nUse the OTP given below to verify your email ID.\n\t\t" + \
86                   str(otp)
87         mail.send(msg)
88         return render_template('verification.html')
89
90     elif ("regmail" in session):
91         if request.method == 'GET':
92             otp = randint(900000, 999999)
93             msg = Message(subject='OTP', sender='hackjacks@gmail.com',
94                           recipients=[session['regmail']])
95             msg.body = "You have succesfully registered for Hire Me!\nUse the OTP given below to verify your email ID.\n\t\t" + \
96                       str(otp)
97             mail.send(msg)
98             return render_template('verification.html', resendmsg="OTP has been resent")
99         else:
100             return redirect('/')
101
102
103 @app.route('/validate', methods=['POST', 'GET'])
104 def validate():
105     if ('regmail' in session):
106         global first_name
107         global last_name
108         global password
109         user_otp = request.form['otp']
110         if otp == int(user_otp):
111             insert_sql = "INSERT INTO User(first_name,last_name,email,password) VALUES (?, ?, ?, ?)"
112             prep_stmt = ibm_db.prepare(connection, insert_sql)
113             ibm_db.bind_param(prepare_stmt, 1, first_name)
114             ibm_db.bind_param(prepare_stmt, 2, last_name)
115             ibm_db.bind_param(prepare_stmt, 3, session['regmail'])
116             ibm_db.bind_param(prepare_stmt, 4, password)
117             ibm_db.execute(prepare_stmt)
118             return render_template('signin.html')
119         else:
```

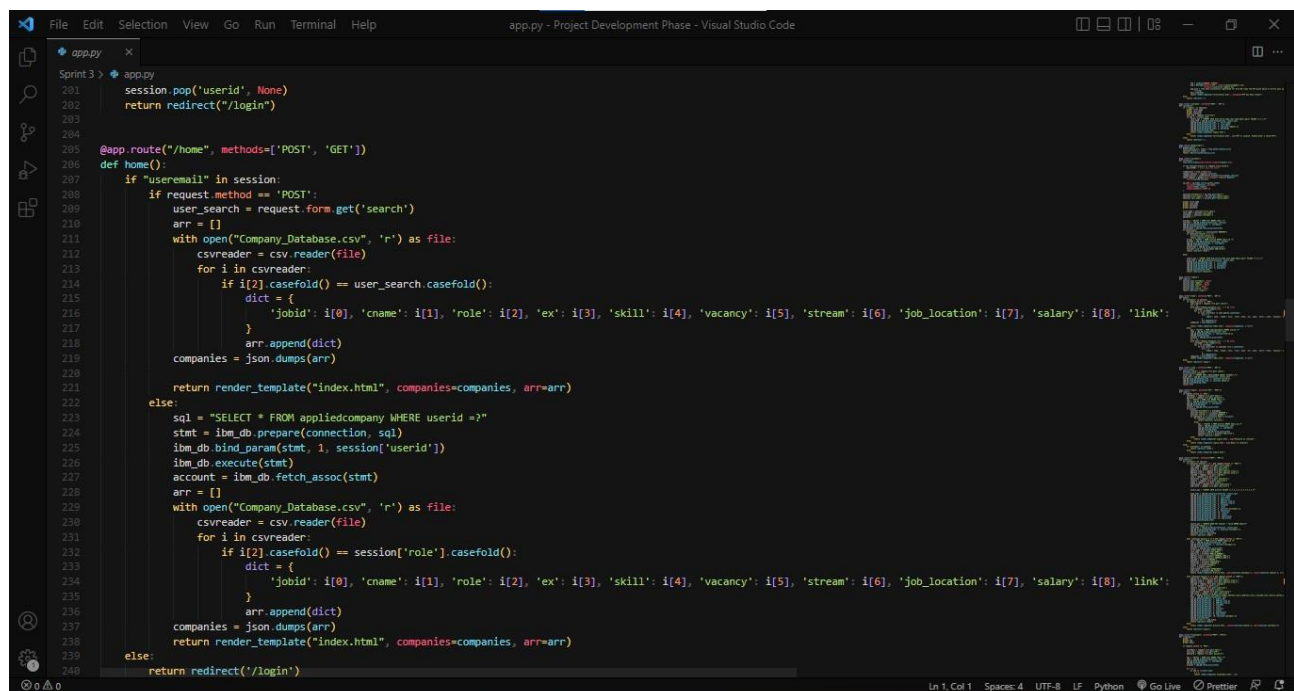
```
File Edit Selection View Go Run Terminal Help app.py - Project Development Phase - Visual Studio Code

app.py
Sprint 3 > app.py
120         return render_template('verification.html', msg="OTP is invalid. Please enter a valid OTP")
121     else:
122         return redirect('/')
123
124
125 @app.route("/googlelogin")
126 def googlelogin():
127     authorization_url, state = flow.authorization_url()
128     session["state"] = state
129     return redirect(authorization_url)
130
131
132 @app.route("/callback")
133 def callback():
134     flow.fetch_token(authorization_response=request.url)
135
136     if not session["state"] == request.args["state"]:
137         abort(500) # State does not match!
138
139     credentials = flow.credentials
140     request_session = requests.session()
141     cached_session = cachecontrol.CacheControl(request_session)
142     token_request = google.auth.transport.requests.Request(
143         session=cached_session)
144
145     id_info = id_token.verify_oauth2_token(
146         id_token=credentials._id_token,
147         request=token_request,
148         audience=GOOGLE_CLIENT_ID
149     )
150
151     session["useremail"] = id_info.get("email")
152     session["first_name"] = id_info.get("given_name")
153     session["last_name"] = id_info.get("family_name")
154
155     global first_name
156     global last_name
157     global useremail
158     global password
```

Cloud Application Development Skill / Job Recommender Application



```
159 first_name = session['first_name']
160 last_name = session['last_name']
161 useremail = session['useremail']
162 password = ""
163
164
165 usersql = "SELECT * FROM User WHERE email =?"
166 userstmt = ibm_db.prepare(connection, usersql)
167 ibm_db.bind_param(userstmt, 1, useremail)
168 ibm_db.execute(userstmt)
169 useraccount = ibm_db.fetch_assoc(userstmt)
170 if useraccount:
171     session['newuser'] = useraccount['NEWUSER']
172     if (session['newuser'] == 1):
173         print(session['newuser'])
174         return redirect('/profile')
175     prosql = "SELECT * FROM profile WHERE email_id =?"
176     prostmt = ibm_db.prepare(connection, prosql)
177     ibm_db.bind_param(prostmt, 1, useremail)
178     ibm_db.execute(prostmt)
179     proaccount = ibm_db.fetch_assoc(prostmt)
180     session['role'] = proaccount['JOB_TITLE']
181     return redirect('/home')
182
183 else:
184
185     insert_sql = "INSERT INTO User(first_name,last_name,email,pass) VALUES (?,?,?,?)"
186     prep_stmt = ibm_db.prepare(connection, insert_sql)
187     ibm_db.bind_param(prepare_stmt, 1, first_name)
188     ibm_db.bind_param(prepare_stmt, 2, last_name)
189     ibm_db.bind_param(prepare_stmt, 3, useremail)
190     ibm_db.bind_param(prepare_stmt, 4, password)
191     ibm_db.execute(prepare_stmt)
192     return redirect("/profile")
193
194
195 @app.route("/logout")
196 def logout():
197     session.pop('useremail', None)
198     session.pop('regmail', None)
```



```
281 session.pop('userid', None)
282 return redirect("/login")
283
284
285 @app.route("/home", methods=['POST', 'GET'])
286 def home():
287     if "useremail" in session:
288         if request.method == "POST":
289             user_search = request.form.get('search')
290             arr = []
291             with open("Company_Database.csv", 'r') as file:
292                 csvreader = csv.reader(file)
293                 for i in csvreader:
294                     if i[2].casefold() == user_search.casefold():
295                         dict = {
296                             'jobid': i[0], 'cname': i[1], 'role': i[2], 'ex': i[3], 'skill': i[4], 'vacancy': i[5], 'stream': i[6], 'job_location': i[7], 'salary': i[8], 'link':
297                         }
298                         arr.append(dict)
299             companies = json.dumps(arr)
300             return render_template("index.html", companies=companies, arr=arr)
301         else:
302             sql = "SELECT * FROM appliedcompany WHERE userid =?"
303             stmt = ibm_db.prepare(connection, sql)
304             ibm_db.bind_param(stmt, 1, session['userid'])
305             ibm_db.execute(stmt)
306             account = ibm_db.fetch_assoc(stmt)
307             arr = []
308             with open("Company_Database.csv", 'r') as file:
309                 csvreader = csv.reader(file)
310                 for i in csvreader:
311                     if i[2].casefold() == session['role'].casefold():
312                         dict = {
313                             'jobid': i[0], 'cname': i[1], 'role': i[2], 'ex': i[3], 'skill': i[4], 'vacancy': i[5], 'stream': i[6], 'job_location': i[7], 'salary': i[8], 'link':
314                         }
315                         arr.append(dict)
316             companies = json.dumps(arr)
317             return render_template("index.html", companies=companies, arr=arr)
318         else:
319             return redirect('/login')
```

Cloud Application Development Skill / Job Recommender Application

```
File Edit Selection View Go Run Terminal Help app.py - Project Development Phase - Visual Studio Code

app.py
Sprint 3 > app.py
243 @app.route('/like', methods=['POST', 'GET'])
244 def store_like():
245     session['jobid'] = request.form.get('jobid')
246     print(session['jobid'])
247     insert_sql = "INSERT INTO LIKES(USERID,JOBIID) VALUES(?,?)"
248     prep_stmt = ibm_db.prepare(connection, insert_sql)
249     ibm_db.bind_param(prepare_stmt, 1, session['userid'])
250     ibm_db.bind_param(prepare_stmt, 2, session['jobid'])
251     ibm_db.execute(prepare_stmt)
252     return None
253
254
255 @app.route("/login", methods=['GET', 'POST'])
256 def login():
257     if request.method == 'POST':
258         useremail = request.form.get('email')
259         password = request.form.get('password')
260         sql = "SELECT * FROM user WHERE email=?"
261         stmt = ibm_db.prepare(connection, sql)
262         ibm_db.bind_param(stmt, 1, useremail)
263         ibm_db.execute(stmt)
264         account = ibm_db.fetch_assoc(stmt)
265         if account:
266             session['useremail'] = useremail
267             session['newuser'] = account['NEWUSER']
268             session['userid'] = account['USERID']
269             if (password == str(account['PASS']).strip()):
270                 if (session['newuser'] == 1):
271                     return redirect('/profile')
272                 else:
273                     sql = "SELECT * FROM profile WHERE email_id=?"
274                     stmt = ibm_db.prepare(connection, sql)
275                     ibm_db.bind_param(stmt, 1, useremail)
276                     ibm_db.execute(stmt)
277                     account = ibm_db.fetch_assoc(stmt)
278                     session['role'] = account['JOB_TITLE']
279                     return redirect('/home')
280             else:
281                 return render_template('signin.html', msg="Password is invalid")
282     else:
```

```
File Edit Selection View Go Run Terminal Help app.py - Project Development Phase - Visual Studio Code

app.py
Sprint 3 > app.py
288     return render_template('signin.html')
289
290
291 @app.route("/profile", methods=['POST', 'GET'])
292 def profile():
293     if "useremail" in session:
294         if (session['newuser'] == 1 and request.method == 'POST'):
295             first_name = request.form.get('first_name')
296             last_name = request.form.get('last_name')
297             mobile_no = request.form.get('mobile_no')
298             address_line_1 = request.form.get('address_line_1')
299             address_line_2 = request.form.get('address_line_2')
300             zipcode = request.form.get('zipcode')
301             city = request.form.get('city')
302             education = request.form.get('education')
303             country = request.form.get('countries')
304             state = request.form.get('states')
305             experience = request.form.get('experience')
306             job_title = request.form.get('job_title')
307
308             insert_sql = "INSERT INTO profile VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?)"
309
310             prep_stmt = ibm_db.prepare(connection, insert_sql)
311             ibm_db.bind_param(prepare_stmt, 1, first_name)
312             ibm_db.bind_param(prepare_stmt, 2, last_name)
313             ibm_db.bind_param(prepare_stmt, 3, mobile_no)
314             ibm_db.bind_param(prepare_stmt, 4, address_line_1)
315             ibm_db.bind_param(prepare_stmt, 5, address_line_2)
316             ibm_db.bind_param(prepare_stmt, 6, zipcode)
317             ibm_db.bind_param(prepare_stmt, 7, city)
318             ibm_db.bind_param(prepare_stmt, 8, session['useremail'])
319             ibm_db.bind_param(prepare_stmt, 9, education)
320             ibm_db.bind_param(prepare_stmt, 10, country)
321             ibm_db.bind_param(prepare_stmt, 11, state)
322             ibm_db.bind_param(prepare_stmt, 12, experience)
323             ibm_db.bind_param(prepare_stmt, 13, job_title)
324             ibm_db.execute(prepare_stmt)
325
326             insert_sql = "UPDATE USER SET newuser = false WHERE email=?"
327             session['newuser'] = 0
```


Cloud Application Development Skill / Job Recommender Application

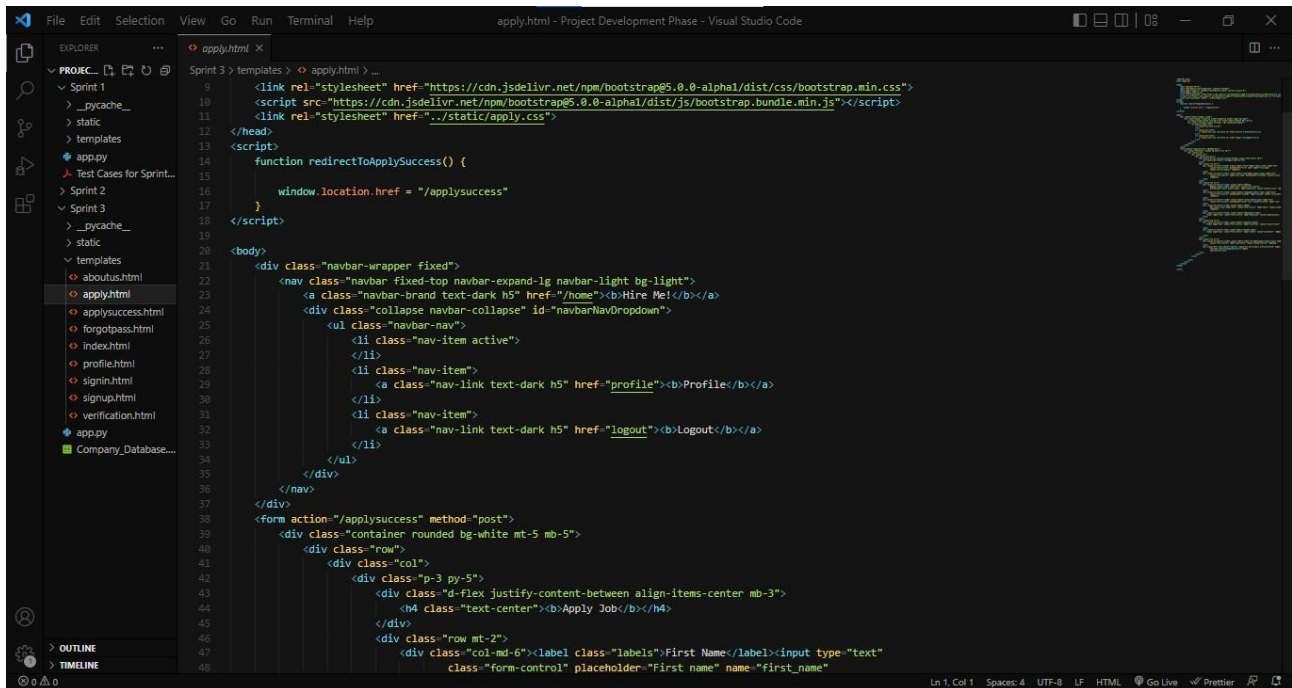
```
app.py - Project Development Phase - Visual Studio Code

333
334
335 elif (session['newuser'] == 0 and request.method == "GET"):
336     sql = "SELECT * FROM profile WHERE email_id=?"
337     stmt = ibm_db.prepare(connection, sql)
338     ibm_db.bind_param(stmt, 1, session['useremail'])
339     ibm_db.execute(stmt)
340     account = ibm_db.fetch_assoc(stmt)
341     first_name = account['FIRST_NAME']
342     last_name = account['LAST_NAME']
343     mobile_no = account['MOBILE_NUMBER']
344     address_line_1 = account['ADDRESS_LINE_1']
345     address_line_2 = account['ADDRESS_LINE_2']
346     zipcode = account['ZIPCODE']
347     education = account['EDUCATION']
348     countries = account['COUNTRY']
349     states = account['STATE']
350     city = account['CITY']
351     experience = account['EXPERIENCE']
352     job_title = account['JOB_TITLE']
353     return render_template('profile.html', email=session['useremail'], newuser=session['newuser'], first_name=first_name, last_name=last_name, address_line_1=address_line_1, address_line_2=address_line_2, zipcode=zipcode, education=education, countries=countries, states=states, city=city, experience=experience, job_title=job_title)
354
355 elif (session['newuser'] == 0 and request.method == "POST"):
356     mobile_no = request.form.get('mobile_no')
357     address_line_1 = request.form.get('address_line_1')
358     address_line_2 = request.form.get('address_line_2')
359     zipcode = request.form.get('zipcode')
360     city = request.form.get('city')
361     country = request.form.get('countries')
362     state = request.form.get('states')
363     experience = request.form.get('experience')
364     job_title = request.form.get('job_title')
365     sql = "UPDATE profile SET (mobile number,address_line_1,address_line_2,zipcode,city,country,state,experience,job_title)=(?,?,?,?,?,?,?,?,?) WHERE email_id=?"
366     stmt = ibm_db.prepare(connection, sql)
367     ibm_db.bind_param(stmt, 1, mobile_no)
368     ibm_db.bind_param(stmt, 2, address_line_1)
369     ibm_db.bind_param(stmt, 3, address_line_2)
370     ibm_db.bind_param(stmt, 4, zipcode)
371     ibm_db.bind_param(stmt, 5, city)
372     ibm_db.bind_param(stmt, 6, country)
373     ibm_db.bind_param(stmt, 7, state)
374     ibm_db.execute(stmt)
```

```
app.py - Project Development Phase - Visual Studio Code

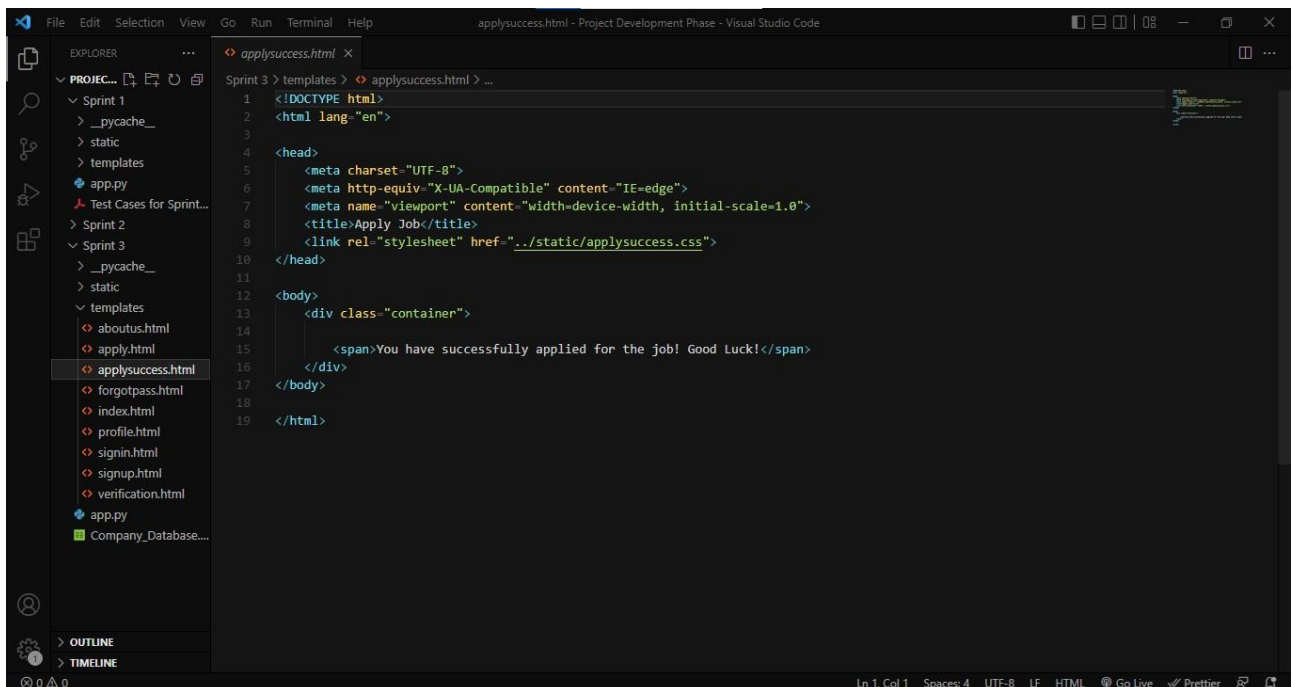
375
376 ibm_db.bind_param(stmt, 10, session['useremail'])
377 ibm_db.execute(stmt)
378 session['role'] = job_title
379 return redirect("/home")
380
381 else:
382     return render_template('profile.html', newuser=session['newuser'], email=session['useremail'])
383
384 else:
385     return redirect("/login")
386
387 @app.route("/forgotpass", methods=['POST', 'GET'])
388 def forgotpass():
389     global i
390     global otp
391     global email
392
393     if request.method == 'POST':
394         useremail = request.form.get('email')
395         user_otp = request.form.get('OTP')
396         password = request.form.get('password')
397
398         sql = "SELECT * FROM User WHERE email=?"
399         stmt = ibm_db.prepare(connection, sql)
400         ibm_db.bind_param(stmt, 1, useremail)
401         ibm_db.execute(stmt)
402         account = ibm_db.fetch_assoc(stmt)
403
404         if i == 1:
405             if otp == int(user_otp):
406                 i = 2
407                 return render_template('forgotpass.html', i=i)
408             else:
409                 return render_template('forgotpass.html', msg="OTP is invalid. Please enter a valid OTP", i=i)
410
411         elif i == 2:
412             sql = "UPDATE USER SET pass=? WHERE email=?"
413             stmt = ibm_db.prepare(connection, sql)
414             ibm_db.bind_param(stmt, 1, password)
415             ibm_db.bind_param(stmt, 2, email)
416             ibm_db.execute(stmt)
```

apply.html :



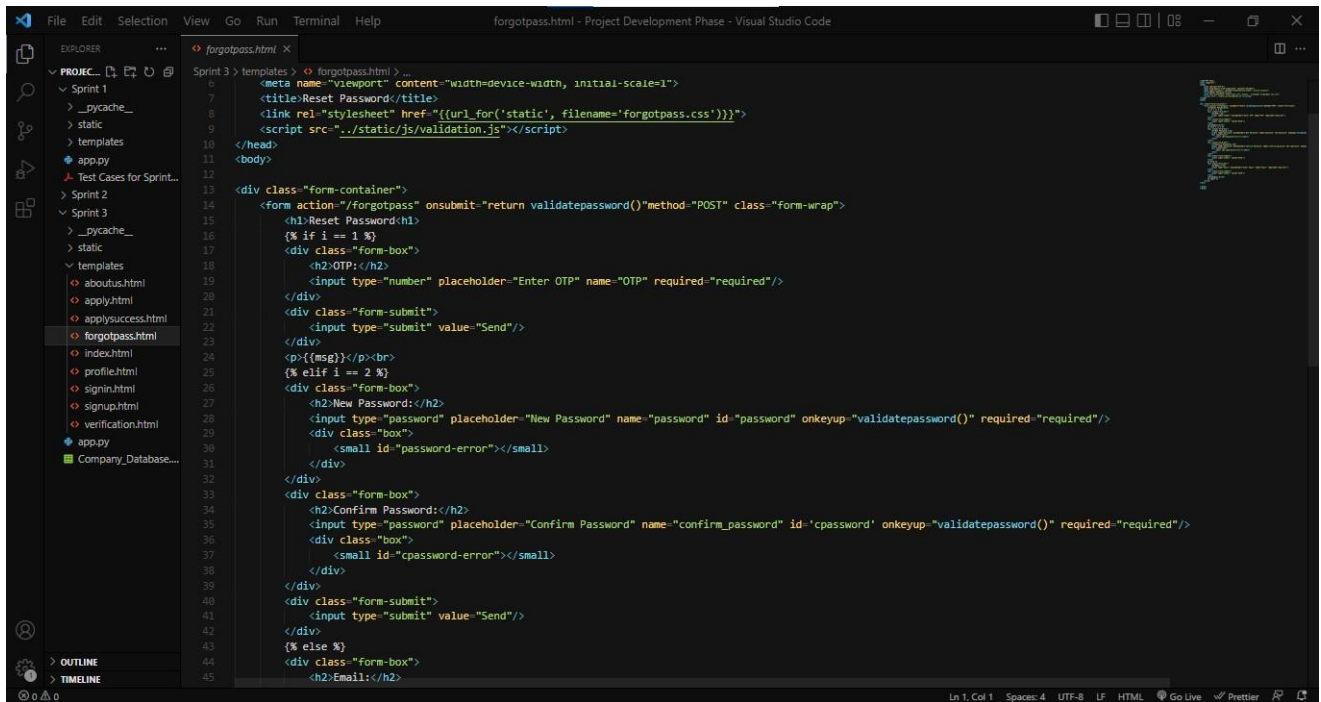
```
9 <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-alpha1/dist/css/bootstrap.min.css">
10 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-alpha1/dist/js/bootstrap.bundle.min.js"></script>
11 <link rel="stylesheet" href="../static/apply.css">
12 </head>
13 <script>
14     function redirectToApplySuccess() {
15         window.location.href = "/appsuccess"
16     }
17 </script>
18 <body>
19     <div class="navbar-wrapper fixed">
20         <nav class="navbar fixed-top navbar-expand-lg navbar-light bg-light">
21             <a class="navbar-brand text-dark h5" href="/home"><b>Hire Me</b></a>
22             <div class="collapse navbar-collapse" id="navbarNavDropdown">
23                 <ul class="navbar-nav">
24                     <li class="nav-item active">
25                         <li class="nav-item">
26                             <a class="nav-link text-dark h5" href="/profile"><b>Profile</b></a>
27                         </li>
28                     <li class="nav-item">
29                         <a class="nav-link text-dark h5" href="/logout"><b>Logout</b></a>
30                     </li>
31                 </ul>
32             </div>
33         </nav>
34     </div>
35     <form action="/appsuccess" method="post">
36         <div class="container rounded bg-white mt-5 mb-5">
37             <div class="row">
38                 <div class="col">
39                     <div class="p-3 py-5">
40                         <div class="d-flex justify-content-between align-items-center mb-3">
41                             <h4 class="text-center"><b>Apply Job</b></h4>
42                         </div>
43                         <div class="row mt-2">
44                             <div class="col-md-6">
45                                 <label class="labels">First Name</label><input type="text"
46                                     class="form-control" placeholder="First name" name="first_name">
47                             </div>
48                         </div>
49                     </div>
50                 </div>
51             </div>
52         </div>
53     </form>
54 </body>
```

appsuccess.html



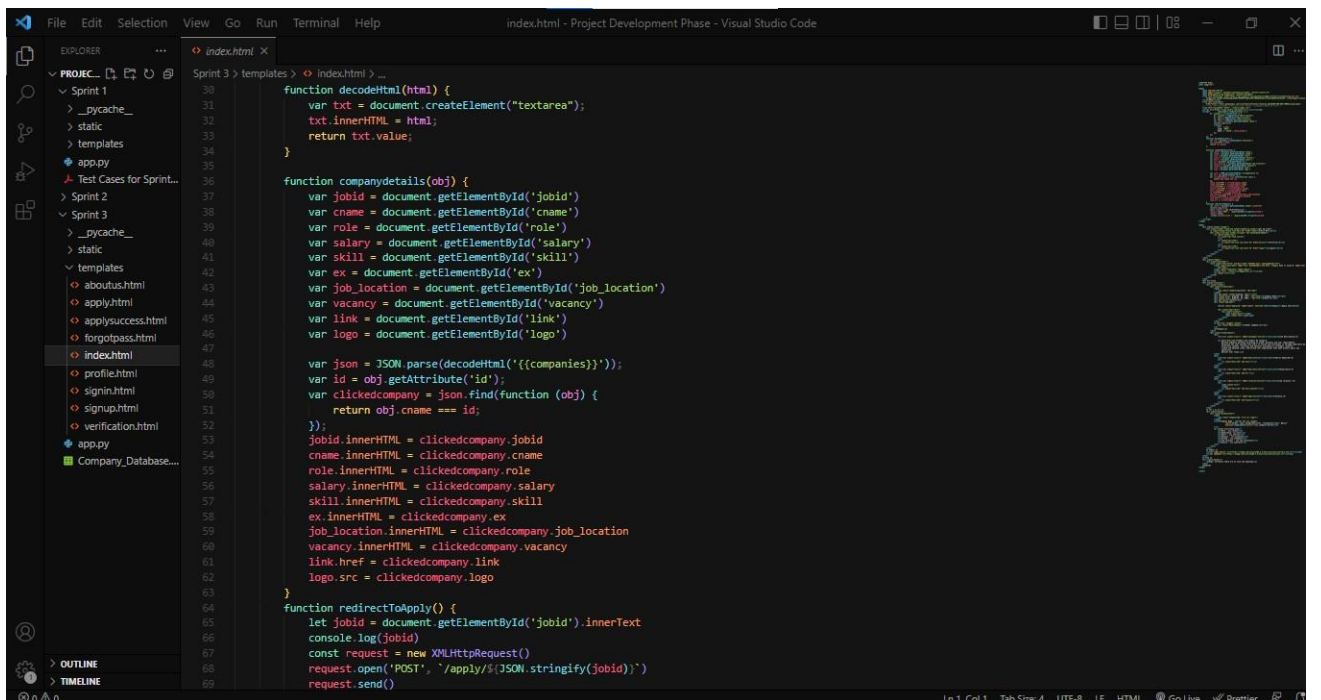
```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5     <meta charset="UTF-8">
6     <meta http-equiv="X-UA-Compatible" content="IE=edge">
7     <meta name="viewport" content="width=device-width, initial-scale=1.0">
8     <title>Apply Job</title>
9     <link rel="stylesheet" href="../static/appsuccess.css">
10 </head>
11
12 <body>
13     <div class="container">
14         <div class="p-3 py-5">
15             <div class="text-center">
16                 <span>You have successfully applied for the job! Good Luck!</span>
17             </div>
18         </div>
19     </div>
20 </body>
```

forgetpass.html



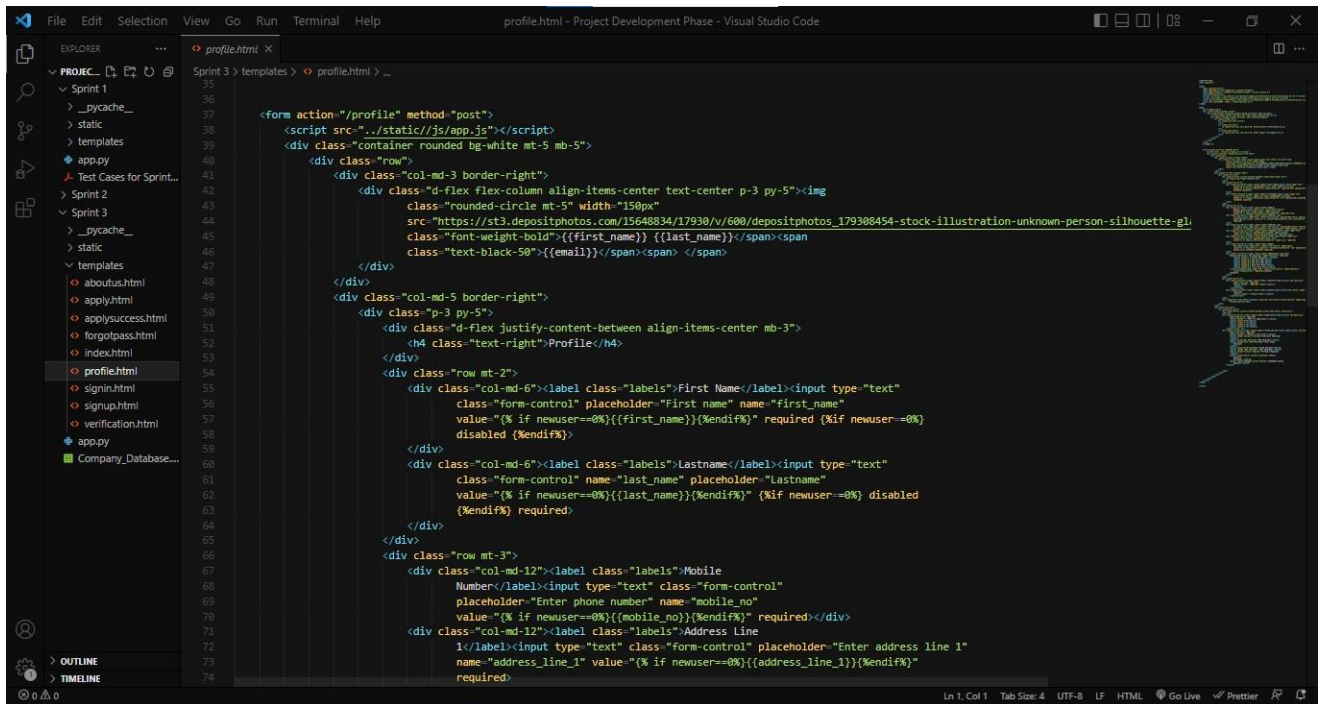
```
1 <!-- forgetpass.html -->
2 <meta name="viewport" content="width=device-width, initial-scale=1">
3 <title>Reset Password</title>
4 <link rel="stylesheet" href="{url_for('static', filename='forgetpass.css')}">
5 <script src="{url_for('static', filename='validation.js')}"></script>
6
7 </head>
8 <body>
9
10 <div class="form-container">
11 <form action="{url_for('forgotpass')}" onsubmit="return validatepassword()" method="POST" class="form-wrap">
12 <h1>Reset Password</h1>
13 <div class="form-box">
14 <h2>OTP</h2>
15 <input type="number" placeholder="Enter OTP" name="OTP" required="required">
16 </div>
17 <div class="form-submit">
18 <input type="submit" value="Send">
19 </div>
20 <p>{{msg}}</p>
21 <div class="form-box">
22 <h2>New Password</h2>
23 <input type="password" placeholder="New Password" name="password" id="password" onkeyup="validatepassword()" required="required">
24 <div class="box">
25 <small id="password-error"></small>
26 </div>
27 <div class="form-box">
28 <h2>Confirm Password</h2>
29 <input type="password" placeholder="Confirm Password" name="confirm_password" id="cpassword" onkeyup="validatepassword()" required="required">
30 <div class="box">
31 <small id="cpassword-error"></small>
32 </div>
33 </div>
34 <div class="form-submit">
35 <input type="submit" value="Send">
36 </div>
37 <div class="form-box">
38 <h2>Email</h2>
39
40 </div>
41 </div>
42
43 </body>
44
45 </html>
```

index.html



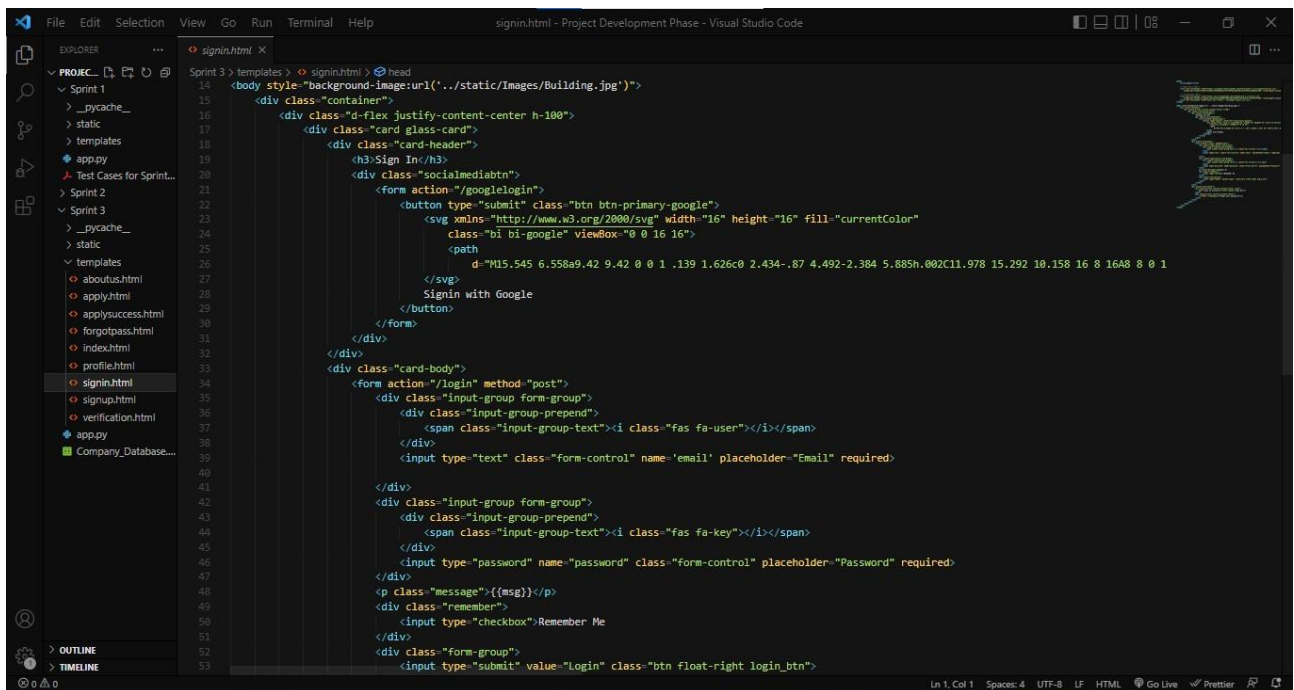
```
1 <!-- index.html -->
2
3 function decodeHtml(html) {
4   var txt = document.createElement('textarea');
5   txt.innerHTML = html;
6   return txt.value;
7 }
8
9 function companydetails(obj) {
10   var jobid = document.getElementById('jobid')
11   var cname = document.getElementById('cname')
12   var role = document.getElementById('role')
13   var salary = document.getElementById('salary')
14   var skill = document.getElementById('skill')
15   var ex = document.getElementById('ex')
16   var job_location = document.getElementById('job_location')
17   var vacancy = document.getElementById('vacancy')
18   var link = document.getElementById('link')
19   var logo = document.getElementById('logo')
20
21   var json = JSON.parse(decodeHtml('{{companies}}'));
22   var id = obj.getAttribute('id');
23   var clickedcompany = json.find(function (obj) {
24     return obj.cname === id;
25   });
26   jobid.innerHTML = clickedcompany.jobid
27   cname.innerHTML = clickedcompany.cname
28   role.innerHTML = clickedcompany.role
29   salary.innerHTML = clickedcompany.salary
30   skill.innerHTML = clickedcompany.skill
31   ex.innerHTML = clickedcompany.ex
32   job_location.innerHTML = clickedcompany.job_location
33   vacancy.innerHTML = clickedcompany.vacancy
34   link.href = clickedcompany.link
35   logo.src = clickedcompany.logo
36 }
37
38 function redirectToApply() {
39   let jobid = document.getElementById('jobid').innerText
40   console.log(jobid)
41   const request = new XMLHttpRequest()
42   request.open('POST', '/apply/' + JSON.stringify(jobid))
43   request.send()
44 }
```


profile.html



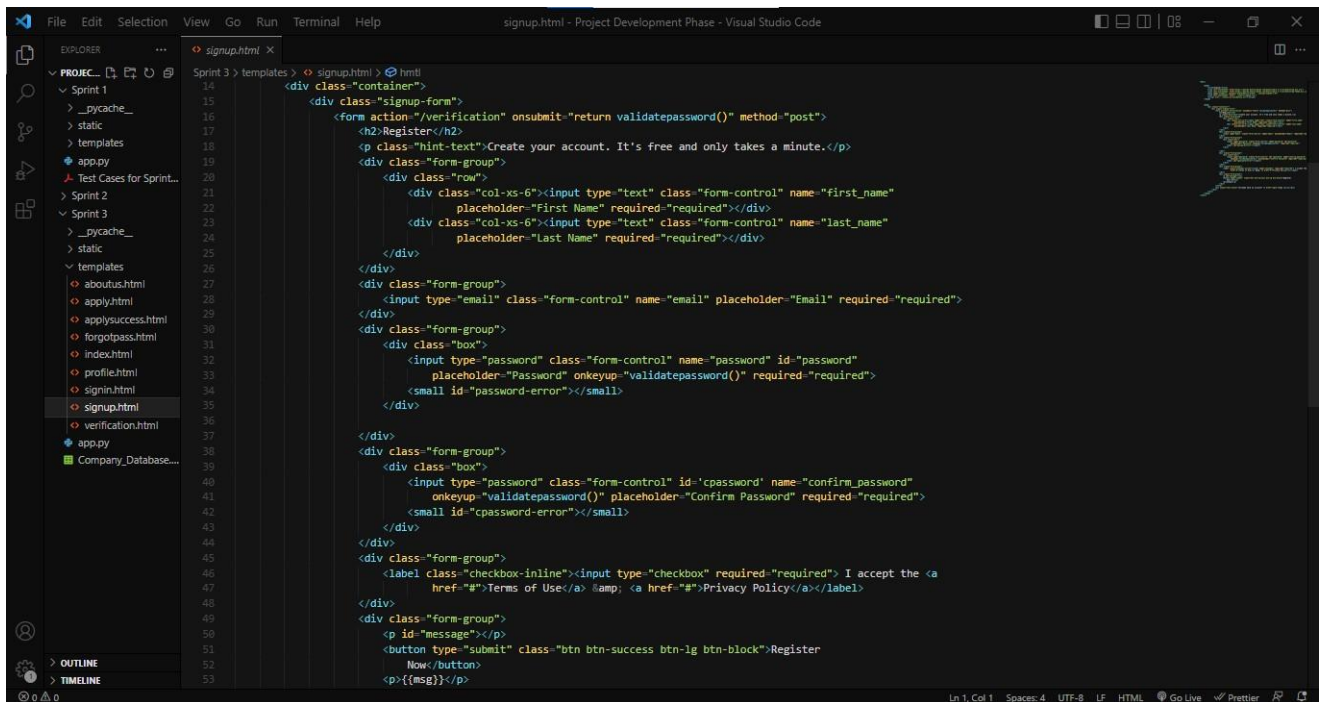
```
13 <form action="/profile" method="post">
14   <script src="../../static/js/app.js"></script>
15   <div class="container rounded bg-white mt-5 mb-5">
16     <div class="row">
17       <div class="col-md-3 border-right">
18         <div class="d-flex flex-column align-items-center text-center p-3 py-5">{{first_name}} </span><span
22           class="text-black-50">{{email}}</span></div>
23       </div>
24       <div class="col-md-5 border-right">
25         <div class="p-3 py-5">
26           <div class="d-flex justify-content-between align-items-center mb-3">
27             <h4 class="text-right">Profile</h4>
28           </div>
29           <div class="row mt-2">
30             <div class="col-md-6"><label class="labels">First Name</label><input type="text"
31               class="form-control" placeholder="First name" name="first_name"
32               value="{% if newuser==0%}&{{first_name}}&{{endif}}&{{endif}} required {%if newuser==0%}
33               disabled {%endif}&{{endif}}
34             </div>
35             <div class="col-md-6"><label class="labels">Lastname</label><input type="text"
36               class="form-control" name="last_name" placeholder="Lastname"
37               value="{% if newuser==0%}&{{last_name}}&{{endif}}&{{endif}} {%if newuser==0%} disabled
38               {%endif}&{{endif}} required
39             </div>
40           </div>
41           <div class="row mt-3">
42             <div class="col-md-12"><label class="labels">Mobile
43               Number</label><input type="text" class="form-control"
44               placeholder="Enter phone number" name="mobile_no"
45               value="{% if newuser==0%}&{{mobile_no}}&{{endif}}&{{endif}} required</div>
46             <div class="col-md-12"><label class="labels">Address Line
47               1</label><input type="text" class="form-control" placeholder="Enter address line 1"
48               name="address_line_1" value="{% if newuser==0%}&{{address_line_1}}&{{endif}}&{{endif}}
49               required
50             </div>
51           </div>
52         </div>
53       </div>
54     </div>
55   </form>
```

signin.html



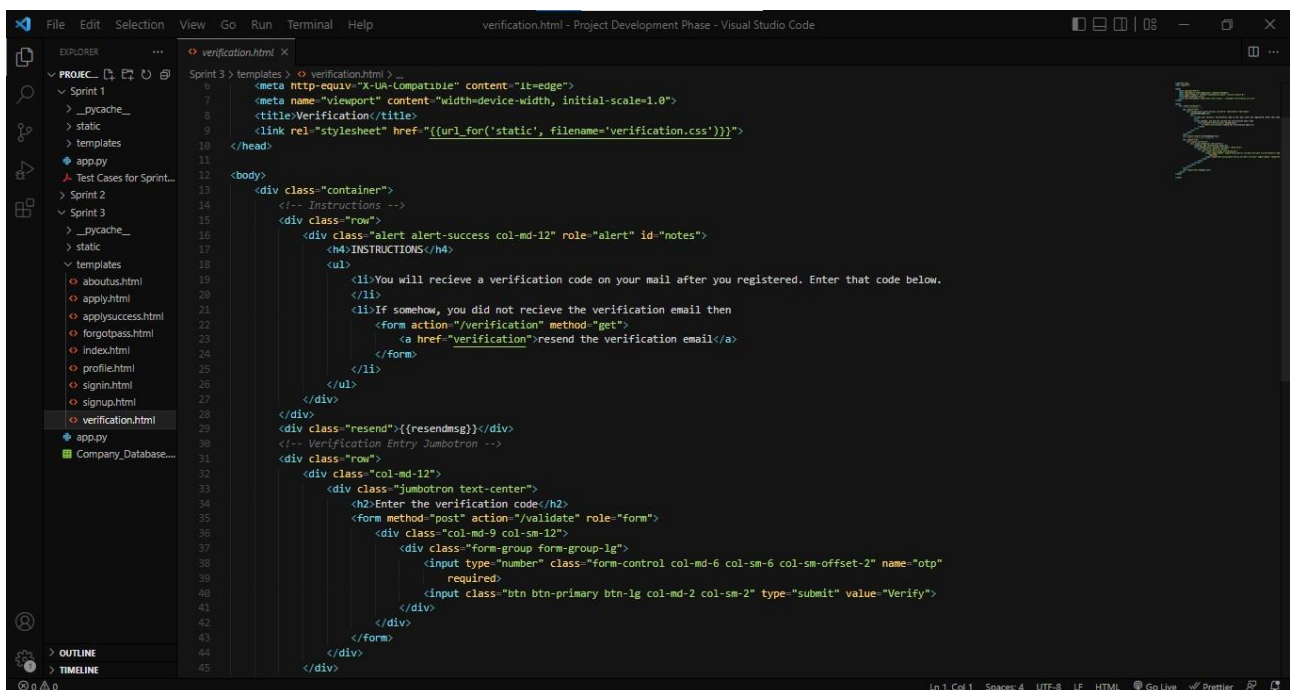
```
13 <body style="background-image:url('../../static/Images/Building.jpg');">
14   <div class="container">
15     <div class="d-flex justify-content-center h-100">
16       <div class="card glass-card">
17         <div class="card-header">
18           <h3>Sign In</h3>
19           <div class="socialmediabtn">
20             <form action="/googlelogin">
21               <button type="submit" class="btn btn-primary-google">
22                 <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor"
23                   class="bi bi-google" viewBox="0 0 16 16">
24                   <path
25                     d="M15.545 6.558a9.42 9.42 0 0 1 .139 1.626c0 2.434-.87 4.492-2.384 5.885h.002c11.978 15.292 10.158 16 8 16a8 0 1
26                   </svg>
27                   Signin with Google
28                 </button>
29               </form>
30             </div>
31           </div>
32         <div class="card-body">
33           <form action="/login" method="post">
34             <div class="input-group form-group">
35               <div class="input-group-prepend">
36                 <span class="input-group-text"><i class="fas fa-user"></i></span>
37               </div>
38               <input type="text" class="form-control" name="email" placeholder="Email" required>
39             </div>
40             <div class="input-group form-group">
41               <div class="input-group-prepend">
42                 <span class="input-group-text"><i class="fas fa-key"></i></span>
43               </div>
44               <input type="password" name="password" class="form-control" placeholder="Password" required>
45             </div>
46             <p class="message">{{msg}}</p>
47             <div class="remember">
48               <input type="checkbox">Remember Me
49             </div>
50             <div class="form-group">
51               <input type="submit" value="Login" class="btn float-right login_btn">
52             </div>
53           </form>
```

signup.html



```
14 <div class="container">
15 <div class="signup-form">
16 <form action="/verification" onsubmit="return validatepassword()" method="post">
17 <h2>Register</h2>
18 <p class="hint-text">Create your account. It's free and only takes a minute.</p>
19 <div class="form-group">
20 <div class="row">
21 <div class="col-xs-6"><input type="text" class="form-control" name="first_name"
22 placeholder="First Name" required="required"></div>
23 <div class="col-xs-6"><input type="text" class="form-control" name="last_name"
24 placeholder="Last Name" required="required"></div>
25 </div>
26 </div>
27 <div class="form-group">
28 <input type="email" class="form-control" name="email" placeholder="Email" required="required">
29 </div>
30 <div class="form-group">
31 <div class="box">
32 <input type="password" class="form-control" name="password" id="password"
33 placeholder="Password" onkeyup="validatepassword()" required="required">
34 <small id="password-error"></small>
35 </div>
36 </div>
37 <div class="form-group">
38 <div class="box">
39 <input type="password" class="form-control" id="cpassword" name="confirm_password"
40 onkeyup="validatepassword()" placeholder="Confirm Password" required="required">
41 <small id="cpassword-error"></small>
42 </div>
43 </div>
44 <div class="form-group">
45 <label class="checkbox-inline"><input type="checkbox" required="required"> I accept the <a
46 href="#">Terms of Use</a> &amp; <a href="#">Privacy Policy</a></label>
47 </div>
48 <div class="form-group">
49 <p id="message"></p>
50 <button type="submit" class="btn btn-success btn-lg btn-block">Register
51 Now</button>
52 <p>{{msg}}</p>
53 </div>
```

verification.html



```
6 <meta http-equiv="X-UA-Compatible" content="IE=edge">
7 <meta name="viewport" content="width=device-width, initial-scale=1.0">
8 <title>Verification</title>
9 <link rel="stylesheet" href="{{url_for('static', filename='verification.css')}}">
10 </head>
11 <body>
12 <div class="container">
13 <!-- Instructions -->
14 <div class="row">
15 <div class="alert alert-success col-md-12" role="alert" id="notes">
16 <h4>INSTRUCTIONS</h4>
17 <ul>
18 <li>You will receive a verification code on your mail after you registered. Enter that code below.
19 </li>
20 <li>If somehow, you did not receive the verification email then
21 <div class="form-group">
22 <form action="/verification" method="get">
23 <a href="/verification">resend the verification email</a>
24 </form>
25 </li>
26 </ul>
27 </div>
28 <div class="resend">{{resendmsg}}</div>
29 <!-- Verification Entry Jumbotron -->
30 <div class="row">
31 <div class="col-md-12">
32 <div class="jumbotron text-center">
33 <h2>Enter the verification code</h2>
34 <form method="post" action="/validate" role="form">
35 <div class="col-md-9 col-sm-12">
36 <div class="form-group form-group-lg">
37 <input type="number" class="form-control col-md-6 col-sm-6 col-sm-offset-2" name="otp"
38 required>
39 </div>
40 <input class="btn btn-primary btn-lg col-md-2 col-sm-2" type="submit" value="Verify">
41 </div>
42 </form>
43 </div>
44 </div>
45 </div>
```

13.2GITHUB & PROJECT DEMO LINK

Github link: <https://github.com/IBM-EPBL/IBM-Project-11360-1659321307>

View the deployed ims-final flask application by clicking the below link:

<http://169.51.203.187:30814/>