

SKILL/JOB RECOMMENDER APPLICATION

TEAM NO : PNT2022TMID43075

COLLEGE NAME : SREE SAKTHI ENGINEERING COLLEGE

DEPARTMENT: COMPUTER SCIENCE & ENGINEERING

TEAM LEADER: SHIBU KM

TEAM MEMBER 1: ARUN B

TEAM MEMBER2:DHANUSIYA S

TEAM MEMBER 3: KUMUTHAM C

TEAM MEMBER 4:THIRU MUGESH B

1 INTRODUCTION :

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.

Project overview :

The phases of the recruitment such as the handling of dates. However, a best fit between job and candidates depends on underlying aspects that are hard to measure. These underlying aspects are a significant reason why information systems have not been extensively used in the area of personnel selection so far.

Purpose :

The intention is to aggregate and recommend appropriate jobs to job seekers, especially in the engineering domain.

2 LITERATURE SURVEY

1. "Students / Job seekers find their desired job based on their Skillset"

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 e- recruiting process and existing recommendation approaches for building personalized recommender systems for candidates/job matching.

2 "Integrating Intelligent CHATBOT for Job recommendation application"

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.

3 "A Study of LinkedIn as an Employment Tool for Job Seeker & Recruiter"

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 LinkedIn 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"

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 Cloud-based file sharing approach is proposed to provide the following services for external data confidentiality, secure data sharing within the group, protect data from unauthorized access of officials within the group and provide time and number of file access to users. Whenever information sharing among a bunch arises, the file owner sends the user upload the file on the application and then shares it using the send API. This creates a safe medium of sharing of files and user in control of the data in the whole process of sharing the files.

REFERENCES

- Adomavicius G, Tuzhilin A (2005). Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Trans. Knowl. Data Eng.* 17(6):734-749
- Barbieri N, Costa G, Manco G, Ortale R (2011). Modeling Item Selection and Relevance for Accurate Recommendations: A Bayesian Approach. In *Proceedings of the fifth ACM conference on Recommender systems (RecSys '11)*, Chicago, Illinois, USA, ACM pp. 21-28.
- Huang Z, Zeng D, Chen H (2007). A Comparative Study of Recommendation Algorithms in ECommerce. *IEEE Intell. Syst.* 22:68-78.
- Felfernig A, Schubert M, Mandl M (2010). Recommendation and Decision Technologies For Requirements Engineering. *RSSE '10*. Cape Town, South Africa, ACM pp. 11-15.

Problem statement definition

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.

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. The proposed recommendation system is tested on an array of test cases with a fully functioning user interface in the form of a web application. It has shown satisfactory results, outperforming the existing systems. It thus testifies to the agenda of quality over quantity.

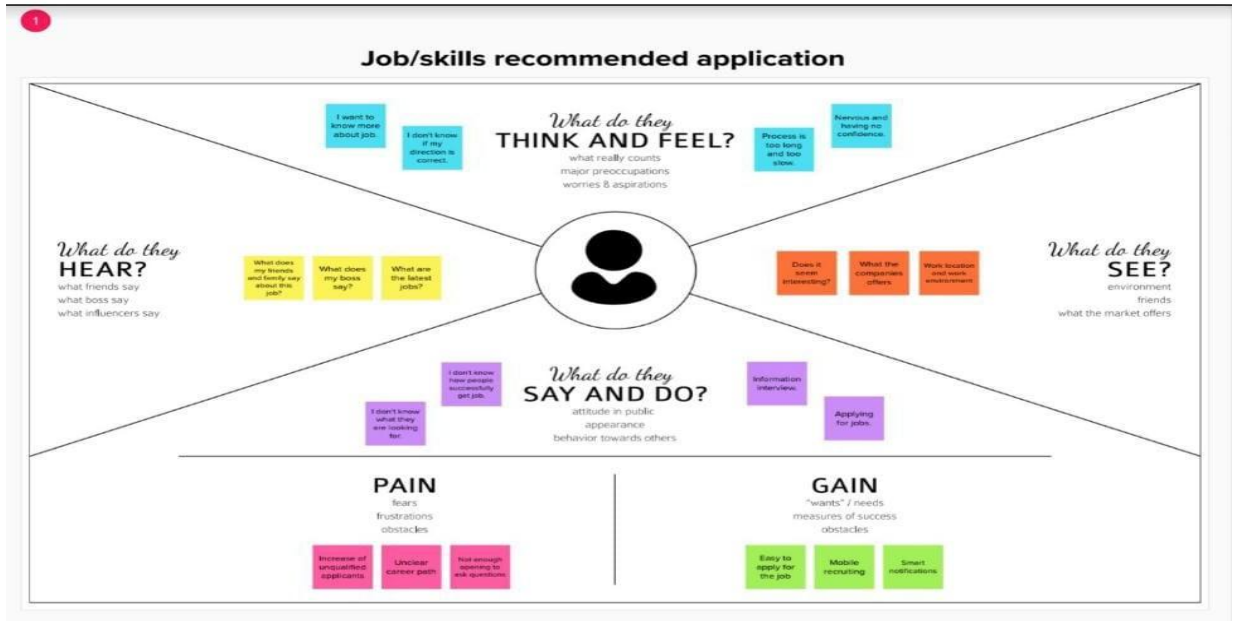
3 IDEATION & PROPOSED SOLUTION

Empathy map canvas

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes.

It is a useful tool to help teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it.

The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.



Ideation & Brainstorming

4

ated notes as you go. Once all
top-like label. If a cluster is
up into smaller sub-groups.

TIP
Add customizable tags to sticky
notes to make it easier to find,
browse, organize, and
categorize important ideas as
they appear on your mural.

Guarantee
Jobs with
needed
salary
Users able
to find a job
they like

Expectations
Trusted jobs
Jobs at user
preferred
location

Importance
If each of these
jobs could get
done without any
difficulty or cost,
which would have
the most positive
impact?

Your team should all be on the same page about what's important
moving forward. Place your ideas on this grid to determine which
ideas are important and which are feasible.

20 minutes

Quick add-ons

- Share the mural**
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.
- Export the mural**
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save to your drive.

Keep moving forward

- Strategy blueprint**
Define the components of a new idea or strategy.
[Open the template](#)
- Customer experience journey map**
Understand customer needs, motivations, and obstacles for an experience.
[Open the template](#)
- Strengths, weaknesses, opportunities & threats**
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
[Open the template](#)

[Share template feedback](#)

Proposed solution

S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Having lots of skills but wondering which job will best suit you? Don't need to worry! We have come up with a skill recommender solution through which the fresher or the skilled person can log in and find the jobs by using the 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 user skillset. The user and their information are stored in the Database. An alert is sent when there is an opening based on the user skillset. Users will interact with the

		<p>chatbot and can get the recommendations based on their skills. We can use a job search API to get the current job openings in the market which will fetch the data directly from the webpage.</p>
2	Idea / Solution description	<p>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 iii) carried out an evaluation to quantify 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.</p>
3	Novelty / Uniqueness	<p>The best positions are suggested to any person according to her skills. While the position of known profiles are assumed should be noted that there are usually multiple advisable positions corresponding to a set of skills. A recommendation system should return a set of most likely positions and all of them can be equally valid. The recommendation method we use is simply based on representing both positions and profiles as comparable vectors and seeking for each profile the positions</p>

		with the most similar vectors.
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

Problem – Solution Fit Template:

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer's problem. It helps entrepreneurs, marketers and corporate innovators identify behavioral patterns and recognize what would work and why

Purpose:

Solve complex problems in a way that fits the state of your customers.

Succeed faster and increase your solution adoption by tapping into existing mediums and channels of behavior.

Sharpen your communication and marketing strategy with the right triggers and messaging.

Increase touch-points with your company by finding the right problem-behavior fit and building trust by solving frequent annoyances, or urgent or costly problems.

Understand the existing situation in order to improve it for your target group.

5 REQUIREMENT ANALYSIS

FUNCTIONAL REQUIREMENT

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

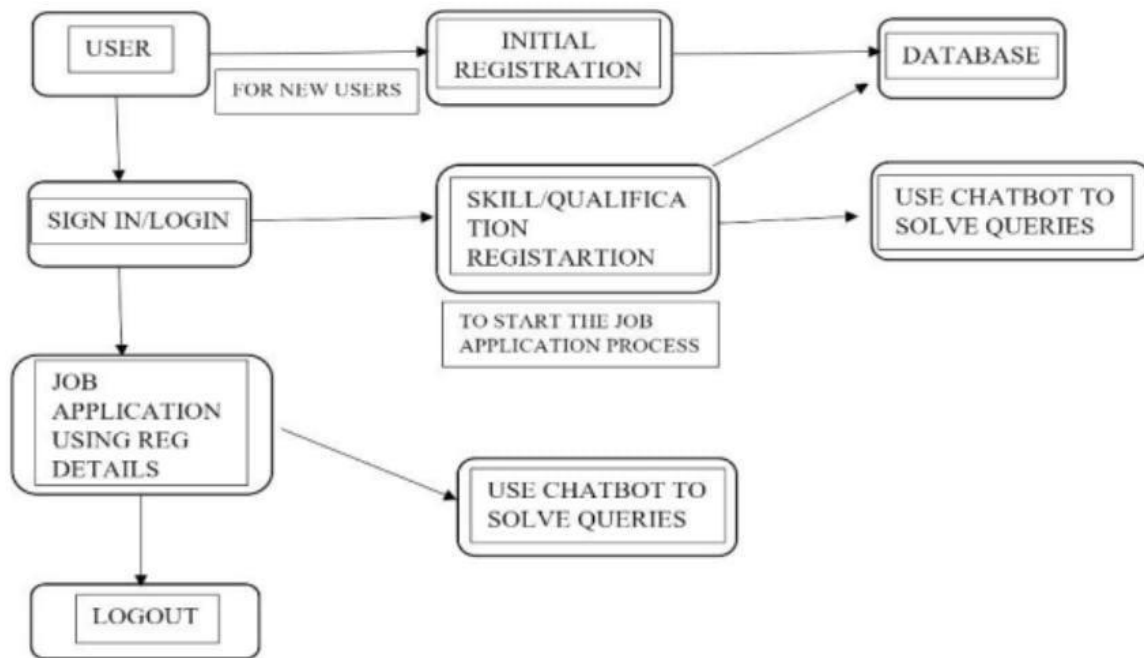
NON-FUNCTIONAL REQUIREMENTS

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
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 anytime during the day
NFR-6	Scalability	Companies can post jobs on the website any time

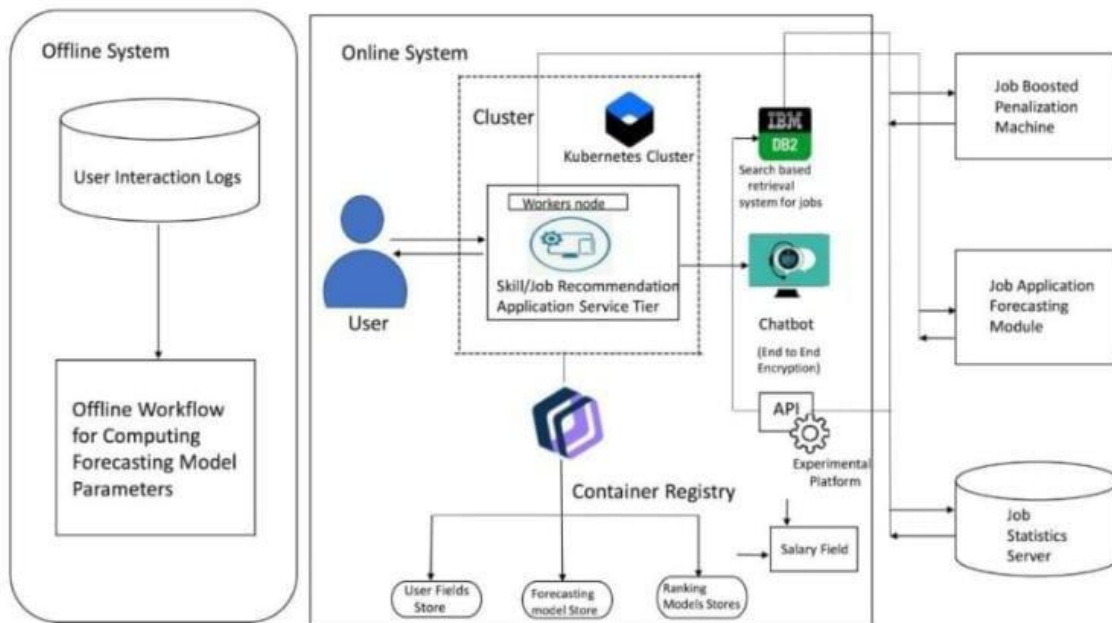
5. PROJECT DESIGN

DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



SOLUTION & TECHINCAL ARCHITECTURE



USER STORIES

User Stories Use the below template to list all the user stories for the product.

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	As a user, I can register for the application through Gmail	MEDIUM	SPRINT-1
	Login	USN-5	As a user, I can log into the application by entering email & password	As a user, I can log into the application by entering email & password	HIGH	SPRINT-1
	Search	USN-6	As a user, I can search for the desired companies	As a user, I can search for the desired companies	HIGH	SPRINT-2

	Apply	USN-7	As a user, I can apply for a company	As a user, I can apply for a company	HIGH	SPRINT-2
	Review	USN-8	As a user, I can review the company	As a user, I can review the company	MEDIUM	SPRINT-2
admin	Forward	USN-9	As an admin, I must forward the applications to the respective companies	As an admin, I must forward the applications to the respective companies	HIGH	SPRINT-1
	Send confirmation	USN-10	Confirmation mail is sent from the respected company	Confirmation mail is sent from the respected company	HIGH	SPRINT-2
	Manage review	USN-11	As an admin, I must make the reviews appear on the company's profile	As an admin, I must make the reviews appear on the company's profile	LOW	SPRINT-2

6 PROJECT PLANNING & SCHEDULING

Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story No	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	Shibu km, Dhanusiya s, Arun b Kumutham c
Sprint-3		USN-2	As a user register instantly using Gmail	4	Low	Arun b Kumutham c Thirumuge sh
Sprint-1	Login	USN-3	As a user,I can log in to the application by entering my email & password	5	High	Shibu km, Kumutham c Thiru mugesh
Sprint-1	Dashboard	USN-4	As a user I can access the dashboard there able to see jobs and filter the jobs using keywords.	6	High	Dhanusiya s, Arun b Kumutham c
Sprint-3		USN-5	A dashboard which shows applied for jobs	6	Medium	Shibu km, Arun b Thiru mugesh

Sprint-2		USN-6	As a user I can see my profile	4	Medium	Dhanusiyas, Kumuthamc
Sprint-2		USN-7	As a user I can update my profile	4	Medium	Shibu km, Dhanusiyas, Kumutham c
Sprint-1	Apply	USN-8	As a user view and apply for the job successfully	4	Medium	Kumuthamc Thirumugesh
Sprint-3		USN-9	track the status of the jobs through a dashboard or email services	4	Medium	Shibu km, Dhanusiyas
Sprint-3	Email	USN-10	As a user get an email about new jobs	6	High	Shibu km, Kumuthamc Thirumugesh
Sprint-2		USN-11	A user noticed after successfully applied job	6	Medium	Shibu km, Arun b Kumutham c
Sprint-2	Bot	USN-12	A bot is embedded in the webpage it' help to users instant matched skill jobs active	6	High	Dhanusiyas, Arun b Kumuthamc
Sprint-4	deploy	USN-13	Creating Docker image	5	Medium	Dhanusiyas, Arun b

						Kumutham c
Sprint-4		USN-14	Making Ui more interactive	5	Low	Shibu km, Dhanusiya s, Arun b
Sprint-4		USN-15	Upload image to IBM container Registry	5	Medium	Shibu km, Dhanusiya s, Arun b Kumutham c
Sprint-4		USN-16	Deploy on Kubernetes	5	Medium	Shibu km, Dhanusiya s, Arun b Kumuthamc Thiru mugesh b

Project Tracker,Velocity & BurndownChart:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint EndDate (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	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022		
Sprint-	20	6 Days	07 Nov	12 Nov 2022		

3			2022			
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022		

VELOCITY:

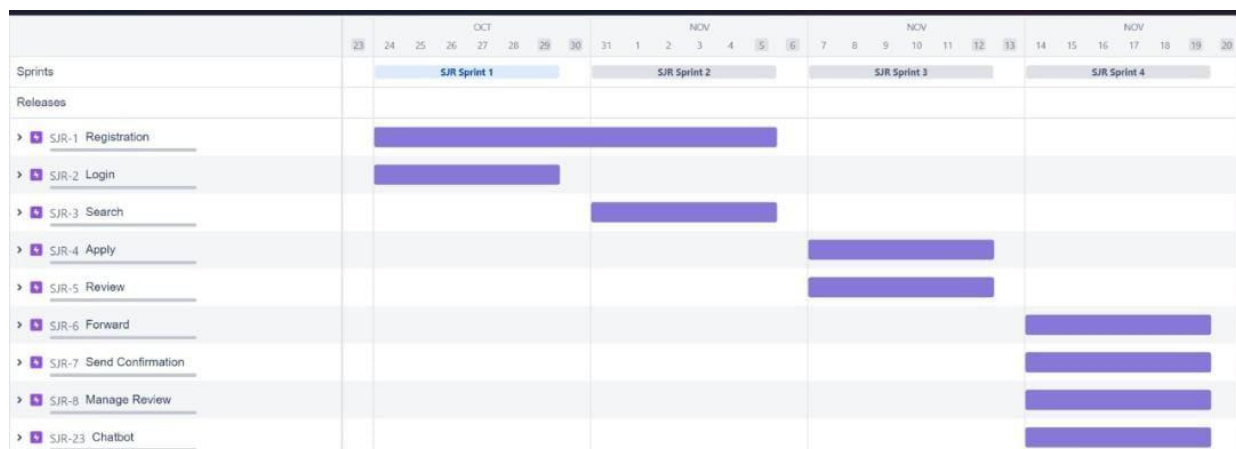
Imagine we have a 6-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{\text{sprint duration}}{\text{velocity}} = \frac{20}{10} = 2$$

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, burndown charts can be applied to any project containing measurable progress over time.

Goal: 60 hours in 5 days



7. CODING & SOLUTION

FEATURE 1

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript / Angular Js / ReactJs etc.
2.	Developing Interface	Developing application for the task	Java / Python
3.	Voice Assistance	Voice commands instead of typing.	IBM Watson STT service .
4.	Chatbot Assistance	Conversational Interface	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	MySQL, NoSQL, etc
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
8.	Machine Learning Model	Purpose of Machine Learning Model	Object Recognition Model, etc.
9.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.

FEATURE 2

S.No	Component	Description	Technology
1.	Open-Source Frameworks	List the open- source frameworks used	Technology of Opensource framework
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	e.g. SHA-256, Encryptions, IAM Controls, OWASP etc.
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Microservices)	Artificial Intelligence (AI) .
4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	balancers, distributed servers etc.) RAID(redundant array of independent disks)
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	DRAM or flash memory

DATABASE

SCHEMASQL

SQL (Structured Query Language) is used to perform operations on the records stored in the database, such as updating records, inserting records, deleting records, creating and modifying database tables, views, etc.

SQL is not a database system, but it is a query language. Suppose you want to perform the queries of SQL language on the stored data in the database. You are required to install any database management system in your systems, for example, Oracle, MySQL, MongoDB, PostgreSQL, SQL Server, DB2, etc. SQL is a short-form of the structured query language, and it is pronounced as S-Q-L or sometimes as See-Quell.

This database language is mainly designed for maintaining the data in relational database management systems. It is a special tool used by data professionals for handling structured data (data which is stored in the form of tables). It is also designed for stream processing in RDBMS. You can easily create and manipulate the database, access and modify the table rows and columns, etc.

This query language became the standard of ANSI in the year of 1986 and ISO in the year of 1987. If you want to get a job in the field of data science, then it is the most important query language to learn. Big enterprises like Facebook, Instagram, and LinkedIn, use SQL for storing the data in the back-end.

8 TESTING

8.1 TEST CASES

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner.

There are various types of tests. Each test type addresses a specific testing requirement. Following this step, a variety of tests are conducted.

Test Cases for Registration Page

TEST CASE S	FEATURE	DESCRIPTION	STEPS TO EXECUTE EXPECTED	RESULTS
TC-001	User Interface	Check all textboxes, checkboxes and buttons	1. Click textboxes, checkboxes and buttons	UI should work properly
TC-002	Required fields	Check the required fields by not filling any data	1. Do not enter any value in the field. 2. Click on the Register button.	A required field message should be displayed
TC-003	Required fields	Check if the user is registered by filling all the required fields	1. Enter valid values in the required fields. 2. Click the Register button.	1. Users should be registered successfully 2. Mail should be sent to the user

TC-004	Required fields	Check if password and confirm password are same	1.Enter different passwords for Password and Confirm Password fields	It should display a message saying that the passwords don't match
TC-004	Email validation	Check if the email is valid	1. Enter Invalid Emails 2. Click on the Register Button.	It should show an invalid email message
TC-005	Email validation	Check all the valid emails	1.Enter Valid Email 2.Click on the Register Button	It should not show any message
TC-006	Email validation	Check if Email already exists in the database.	1.Enter an already registered email. 2.Click Register button	It should say that email already exists

Test Cases for Login Page

TEST CASES	FEATURE	STEPS TO EXECUTE	EXPECTED	RESULTS
TC-001	User Interface	Check all textboxes, checkboxes and buttons	1.Click textboxes, checkboxes and buttons	UI should work properly
TC-002	Required fields	Check the required fields by not filling	1. Do not enter any value in the field. 2. Click	A required field message should be

		any data	on the Login button.	displayed
TC-003	Required fields	Check user should by filling all the required fields	1. Enter valid values in the required fields. 2. Click the Login button.	1. Users should be logged in successfully 2. User should be redirected to home page
TC-004	Email validation	Check if the email is valid	1. Enter Invalid Emails 2. Click on the Login Button.	It should show an invalid email message
TC-005	Required fields	Check if Password is valid	1. Enter Invalid password 2. Click on the Login button	It should show invalid password message

Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application; it is done after the completion of an individual unit before integration.

This is a structural testing that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or System configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields.

Integration tests demonstrate that although the components were individually satisfied, as shown by successively unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problem that arises from the combination of components.

Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input** : identified classes of valid input must be accepted.
- Invalid Input** : identified classes of invalid input must be rejected.
- Function** : identified functions must be exercised.
- Output** : identified classes of application outputs must be exercised.
- Systems/Procedures:** interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identifying Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It has a

purpose. It is used to test areas that cannot be reached from a blackbox level.

Unit Testing

Unit test is usually conducted as part of a combined code and unit test and unit testing phase of the software lifecycle, although it is not uncommon for coding and unit tests to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

All field entries must work properly.

Pages must be activated from the identified link.

The entry screen, messages and responses must not be delayed.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or one step up- software applications at the company level - interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

8.2. USER ACCEPTANCE TESTING:

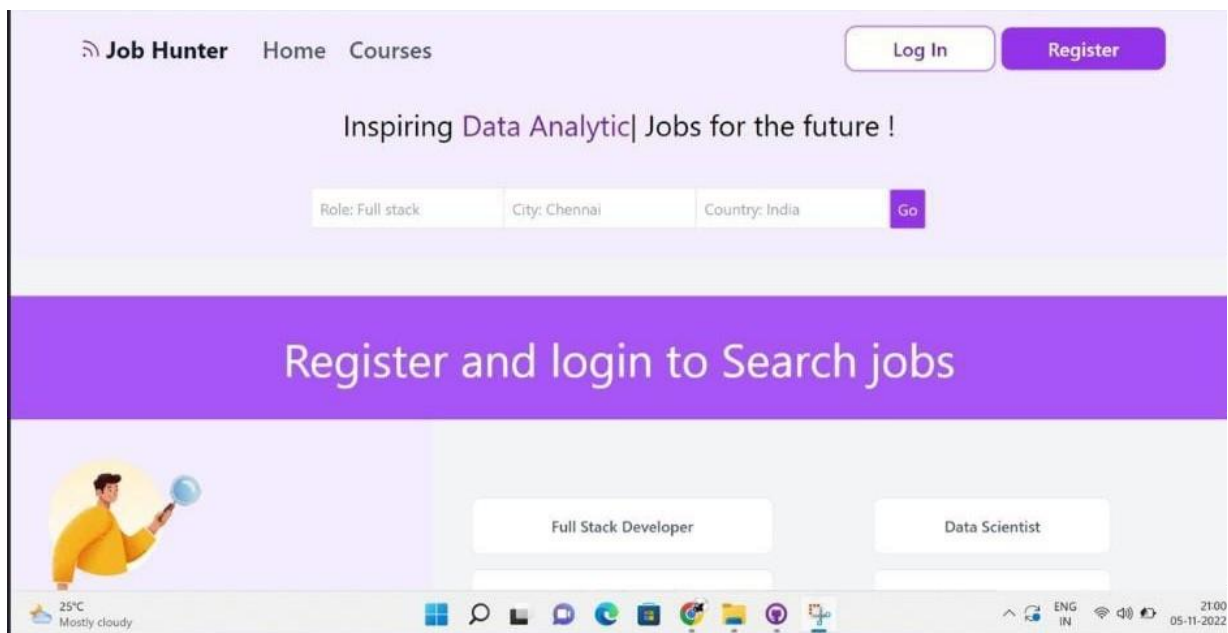
User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

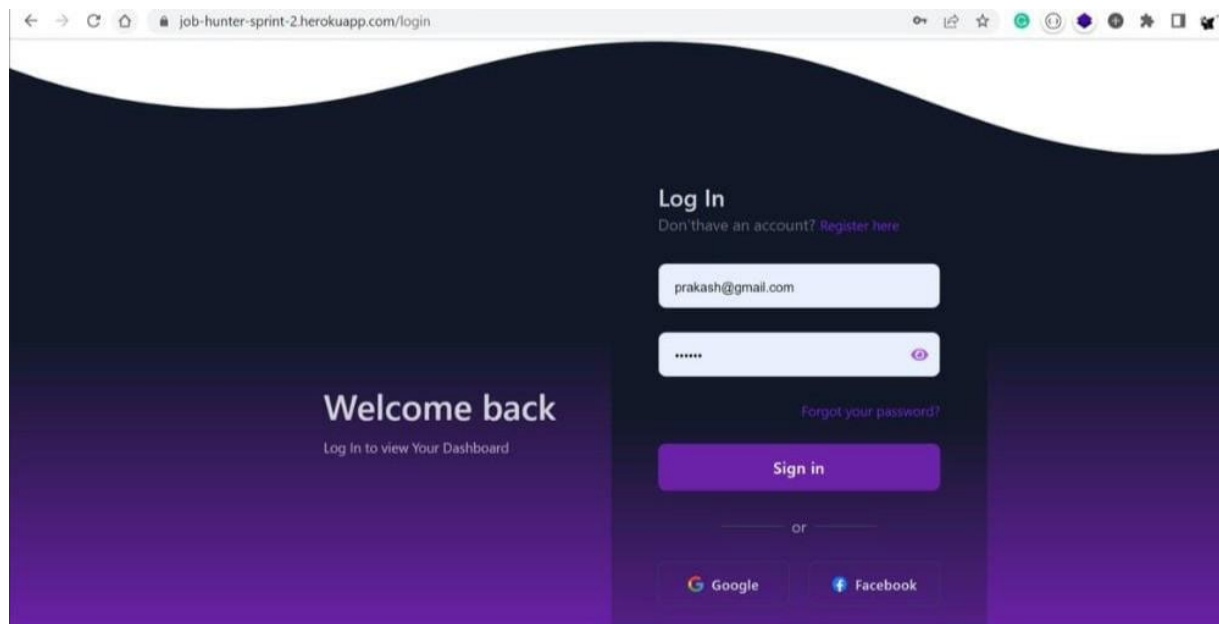
9.RESULTS

9.1 Performance Metrics

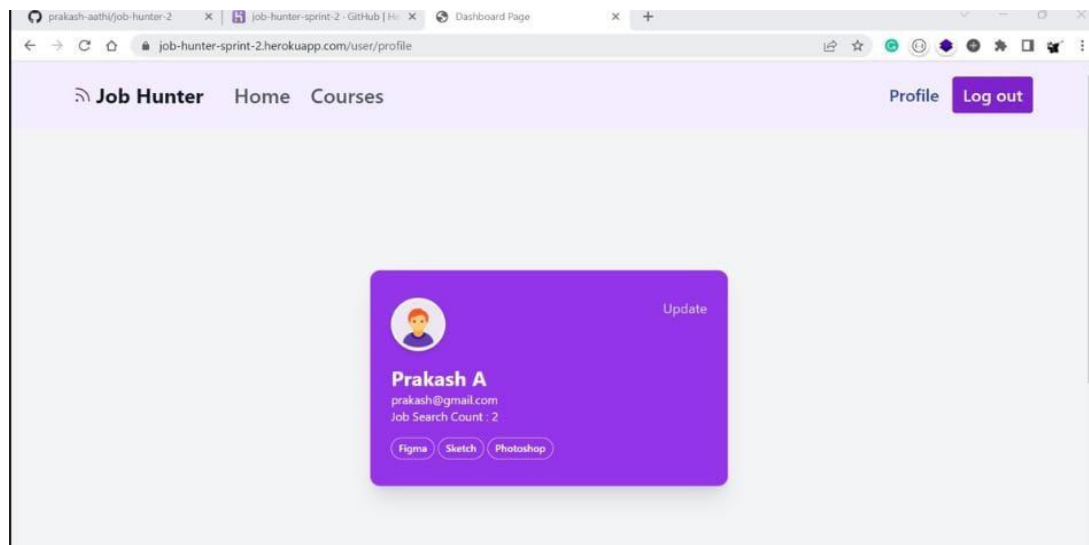
1) Home page



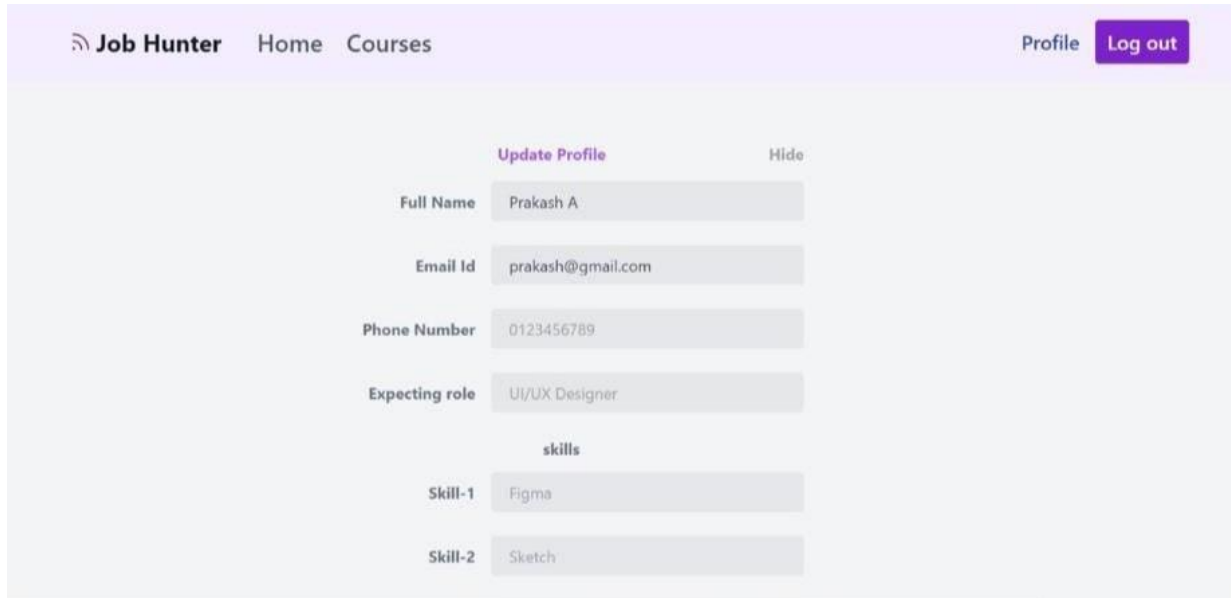
2. login page



3. profile



4. Registration page



The screenshot shows a web interface for 'Job Hunter'. The top navigation bar includes 'Home' and 'Courses' links, and a 'Profile' link with a 'Log out' button. The main content area is titled 'Update Profile' with a 'Hide' button. The form contains the following fields:

Field	Value
Full Name	Prakash A
Email Id	prakash@gmail.com
Phone Number	0123456789
Expecting role	UI/UX Designer
skills	
Skill-1	Figma
Skill-2	Sketch

10) ADVANTAGES AND DISADVANTAGES

Advantages:

The model doesn't need any data about other users, since the recommendations are specific to this user. This makes it easier to scale to a large number of users.

The model can capture the specific interests of a user, and can recommend niche items that very few other users are interested in.

Disadvantages:

Since the feature representation of the items are hand-engineered to some extent, this technique requires a lot of domain knowledge. Therefore, the model can only be as good as the hand-engineered features.

The model can only make recommendations based on existing interests of the user. In other words, the model has limited ability to expand on the users' existing interests.

11. CONCLUSION

In this paper, we proposed a framework for job recommendation task. This framework facilitates the understanding of job recommendation process as well as it allows the use of a variety of text processing and recommendation methods according to the preferences of the job recommender system designer. Moreover, we also contribute making publicly available a new dataset containing job seekers profiles and job vacancies. Future directions of our work will focus on performing a more exhaustive evaluation considering a greater amount of methods and data as well as a comprehensive evaluation of the impact of each professional skill of a job seeker on the received job recommendation.

While the future of work remains unclear, change is inevitable. New technologies, economic crises, and other factors will continue to shift labor demands causing workers to move between jobs. If labor transitions occur efficiently, significant productivity and equity benefits arise at all levels of the labor market [45]; if transitions are slow, or fail, significant costs are borne to both the State and the individual. Therefore, it is in the interests of workers, firms, and governments that labor transitions are efficient and effective. The methods and systems we put forward here could significantly improve the achievement of these goals.

12. FUTURE SCOPE

Our application is not finished yet. There are many rooms for improvement. Some of them will be improved in the future versions

Attracting and much more responsive UI throughout the application

Releasing cross-platform mobile applications

Incorporating automatic replies in the chat columns

Deleting the account whenever customer wishes to

Supporting multi-media in the chat columns

Creating a community for our customers to interact with one another

Call support

Instant SMS alerts

13. APPENDIX

SOURCE CODE:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>{% block title%}{% endblock%}</title>
  <!-- link tailwind css -->
  <script src="https://cdn.tailwindcss.com"></script>
  {% block link%}{% endblock%}
</head>

<body>

  <!-- nav bar start -->
  <div class="bg-gray-200">
    <div class="max-w-6xl px-4 mx-auto border ">
```

```

<div class="flex justify-between">

  <div class="flex space-x-4 ">
    <!-- logo -->
    <div class="">
      <a href="#" class="flex items-center py-5 px-2 text-gray-700 hover:text-gray-900">
        <svg class="h-6 w-6 mr-1 text-red-400" xmlns="http://www.w3.org/2000/svg"
fill="none"
viewBox="0 0 24 24" stroke-width="1.5" stroke="currentColor" class="w-6 h-6">
  <path stroke-linecap="round" stroke-linejoin="round"
d="M12.75 19.5v-.75a7.5 7.5 0 00-7.5-7.5H4.5m0-6.75h.75c.75 0 1.425 .638 1.425
14.25v.75M6 18.75a.75.75 0 11-1.5 0 .75.75 0 011.5 0z" />
</svg>
<span class="font-bold">Job Hunter</span>
</a>
</div>
<!-- primary nav -->
<div class="flex hidden md:flex items-center spcae-x-1 ">
  <a href="{ {url_for('check')}}" class="py-5 px-3 text-gray-700 hover:text-gray-
900">Home</a>
  <a href="{ {url_for('course')}}" class=" py-5 px-3 text-gray-700 hover:text-gray-
900">Courses</a>

    </div>
  </div>
  <!-- sec nav -->
  <div class="flex hidden md:flex items-center space-x-1">
    { % block lgNav % }
    { %endblock% }
  </div>
  <!-- mobile button -->
  <div class="md:hidden flex items-center">
    <button class="mobile-menu-button">
      <svg class="w-6 h-6" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24"
stroke-width="1.5" stroke="currentColor" class="w-6 h-6">
        <path stroke-linecap="round" stroke-linejoin="round" d="M3.75
6.75h16.5M3.75 12h16.5m-16.5 5.25h16.5" />

```



```

</svg>
</button>
</div>

</div>
</div>
<!-- mobile menu -->
<div class="hidden mobile-menu md:hidden">
  <a href="#" class="block py-2 px-4 text-sm hover:bg-gray-200">About</a>
  <a href="#" class="block py-2 px-4 text-sm hover:bg-gray-200">Courses</a>
    { % block mdNav% }
    { %endblock% }
  </div>

</div>
<!-- nav bar scripts -->
<script>
  / grab everything we need
  const btn = document.querySelector('button.mobile-menu-button');const menu =
  document.querySelector(".mobile-menu");

  / add event listener
  btn.addEventListener("click", () => {
    menu.classList.toggle("hidden");
  });
</script>
<!-- nav bar end -->

{ % block body % }
{ % endblock% }

</body>

</html>

{ % block lgNav % }
<a href="{ {url_for('profile')} }" class="py-5 px-3 text-blue-800">Profile</a>

```

```

<a href="{{url_for('logout')}}" class="py-2 px-3 bg-purple-700 text-white hover:bg-purple-300
hover:text-purple-800 rounded transition duration-300 ">Log out</a>
{%endblock%}
{% block mdNav%}
<a href="{{url_for('profile')}}" class="block py-2 px-4 text-sm text-blue-800 hover:bg-gray-
200">Profile</a>
<a href="{{url_for('logout')}}" class="block py-2 px-4 text-sm hover:bg-gray-200">Log
out</a>
{%endblock%}

{% block body %}
<!-- profile card -->
<!--BG-->
<div class="flex justify-center items-center h-screen ">
  <!--Card-->
    <div class="bg-purple-600 flex flex-col w-2/3 md:w-1/3 gap-4 py-8 px-6 rounded-xlshadow-
xl border border-purple-400">
      <!--Avatar-->
      <div class="flex justify-between">
        <div></div>
        <div><button class="text-gray-300 hover:text-white">Edit</button></div>
      </div>
      <div class="text-white">
        <!--Role-->
        <p class="text-sm">UI/UX Designer</p>
        <!--Name-->
        <p class="font-bold text-2xl">{{username}}</p>
        <!--Description-->
        <p class="text-justify text-sm">{{email}}</p>
        <!-- phone number -->
        <p class="text-justify text-sm"></p>
      </div>
      <!--Tools chip-->
      <div class="flex flex-row gap-1 text-xs text-white font-semibold">
        <div class="rounded-full border border-purple-100 py-1 px-2">

```

```

        <span>Figma</span>
    </div>
    <div class="rounded-full border border-purple-100 py-1 px-2">
        <span>Sketch</span>
    </div>
    <div class="rounded-full border border-purple-100 py-1 px-2">
        <span>Photoshop</span>
    </div>
</div>
</div>
</div>
<!-- profile card end -->
{% endblock% }
app = Flask(_name_)
app.secret_key="123"

@app.route('/')def
home():
    ''' Route Home page '''
    return render_template ("index.html")

@app.route("/user/<username>") def
dashboard(username):
    ''' Route Dashboard. When user log in the session is stored if the user logged in itredirect
to
    dashboard else it redirect to login page '''if
('user' in session):
    jobs=jobLoad.fetchDefaultJob()
                                                                    return
render_template('dashboard.html',username=session['user'],jobs=jobs,count="None")
    else:
        return redirect(url_for('login'))
@app.route('/login')
def login():
    ''' Route login Page. when click login page it checks session is active if active itredirects
    dashboard else redirect to login page'''

```

```
if ('user' in session):
    return redirect(url_for("dashboard",username=session['user']['name'])) else:
    return render_template("./auth/login.html")
```

```
@app.route('/logout') def
logout():
    """ Route logout. When use click log out the user session is deleted """
    session.pop('user')
    return redirect(url_for('home'))
```

```
@app.route('/register') def
register():
    """ Route register page. It render register page """return
    render_template("./auth/register.html")
```

```
@app.route('/registerData',methods=["POST",'GET']) def
registerData():
    """ Route register Data. when user filled the register form the details are verified if it'svalid
    redirect
    to home page else it rendered register page with exception message """if
    request.method=='POST':
        name=request.form.get('name')
        email=request.form.get('email')
        password=request.form.get('password')
        error= registerData.checkValid(name,email,password)if
        error != None:
            return render_template("./auth/register.html",error=error) try:
                user=LogAuth.auth.create_user_with_email_and_password(email,password)
                fetch.addData(user['idToken'],name,email)
            except :
                error="You already Registered Please Login with Your Credentials"
                return render_template("./auth/register.html",error=error)
    return redirect(url_for('home')) @app.route('/loginData',methods=["POST","GET"])
```

```
def logindata():  
    """ Route login Data. When user filled the login form and press submit the api passemail,  
    password.
```

```
    The email and password verify if exists in database. If exist fetch the user data andcreate a  
    session"""
```

```
    if request.method=='POST':try:  
        emailEl=request.form['emailEl']  
        passwordEl=request.form['passwordEl']  
        user = LogAuth.auth.sign_in_with_email_and_password(emailEl,passwordEl) userId  
        =(user['localId'])  
        # fetch method ==> fetch user data  
        fetchData=fetch.fetchData(userId)  
        session['user']=fetchData  
    except:  
        error="Your email/password are not matched.."  
        return render_template("./auth/login.html",error=error) return  
        redirect(url_for("dashboard",username=fetchData['name']))
```

```
@app.route('/user/profile') def
```

```
profile():
```

```
    """ Route Profile page. It shows user information stored in session"""const  
    =session['user']  
    return render_template('profile.html',data=const)
```

```
@app.route('/check') def
```

```
check():
```

```
    """ Route check. if user logged in it shows dashboard else it shows landing page """if ('user'  
    in session):  
        return redirect(url_for('dashboard',username=session['user']['name']))else:  
        return redirect(url_for('home'))
```

```
@app.route('/course') def
```

```
course():
```

```
    """ Route course. if logged in it shows profile button else it shows log in button """if ('user'  
    in session):
```

```
return render_template('course.html', see=True )else:
return render_template('course.html', see=False )
```

```
@app.route('/update',methods=["POST","GET"]) def
update():
```

```
    """ Route update. It fetch all details in profile update form and update in database and session """
```

```
    if request.method=='POST':try:
```

```
        id = session['user']['id']
        name = session ['user']['name'] email
        =session['user']['email']
        number=request.form['number'] role
        =request.form['role']
        skill1    =request.form['skill1']
        skill2    =request.form['skill2']
        skill3 =request.form['skill3']
```

```
data={'id':id,'name':name,'email':email,'number':number,'role':role,'skill1':skill1,'skill2':skill2,
      'skill3':skill3,'update':True,'jobFetchCount':session['user']['jobFetchCount']}
profileUpdate.update(id,data)
session['user']=data
except:
    return "user update failed"
return render_template('profile.html',data=data)
```

```
@app.route('/search',methods=['POST']) def
search():
```

```
    """ Route search. Each user gets 3 free search so it check search limit stored in session if it's
    under
```

```
    limit request job details in job api and the job results are passes into dasboard and the results
    also
```

```
    stored in database. meanwhile the Search limit is updated in cloud and session """if
```

```
    request.method=='POST':
```

```
        try:
```

```
            id=session['user']['id']
```

```

count=session['user']['jobFetchCount']
role=request.form['role']
city=request.form['city']
country=request.form['country']
if count < 3:
    # jobLoad.jobapi(id,count,role,city,country)
    jobs=jobLoad.fetchDefaultJob(id+str(session['user']['jobFetchCount']))
    data=session['user']
    data['jobFetchCount']= data['jobFetchCount']+1
    session['user']=data profileUpdate.update(id,data)
return
render_template('dashboard.html',username=session['user'],jobs=jobs,count=data['jobFetchCount']-
1)
    else:
        return "You Used 3 searches If you need more Please Mail to this ID
prakash.ece19@gmail.com"
except:
    return "Search failed"
    return redirect(url_for("dashboard",username=session['user']))

@app.route("/history/<history>") def
history(history):
    ''' Route History. The history of job searches are stored in database. when the userrequest to
see the jobs
and details. it get from database and showed in history page '''error=""
if session['user']['jobFetchCount']!=0:
    print((session['user']['jobFetchCount'])) print(int(history))
    if (session['user']['jobFetchCount'] > int(history):
        print(". .. loaded")
        jobs=jobLoad.fetchDefaultJob(str(session['user']['id'])+(history))
        return render_template("history.html",history=history,jobs=jobs,count=history)else:
            error="No Search History. Search the job results are saved in history"#
            return render_template("history.html")

```

```

        # return "No Search results"
    else:
        error="you don't have search history"
        # return "you not having search history"
    return render_template("history.html",error=error)

@app.route('/user/apply/<x>',methods=['GET']) def
apply(x):
    """ In home latest jobs are shown when user request to show more details about latestjob these
    method fetch
    data about job description,apply link it showed in apply.html"""
    jobDetails = jobDetailsLoad.findJobDetails(x)
    return render_template('apply.html',job=jobDetails[0])

@app.route('/user/applys/<count>/<x>',methods=['GET']) def
applys(count,x):
    """ In search result jobs are stored in database when user request the search history ofjob
    description,apply
    link the applys method fetch from database """ merge =
    session['user']['id']+str(int(count)) jobDetails =
    jobDetailsLoad.findJobDetails(x,merge)
    return render_template('apply.html',job=jobDetails[0])

@app.route("/<job>")
def roleBasedJob(job):
    """ These route method rol based job details from database """# under
    process
    if job=="Full Stack Developer":
        jobs=jobLoad.fetchDefaultJob("defaultJob")
        return "coming soon sprint-3"
    return render_template('dashboard.html',username='Guest',jobs=jobs,count="None")

if __name__ == "__main__":
    app.run('0.0.0.0',port=8080,debug=True)
{% extends 'base.html' %}

{% block title %}

```


Dashboard Page

{% endblock %}

{% block lgNav %}

Profile

<a href="{ {url_for('logout') }}" class="py-2 px-3 bg-purple-700 text-white hover:bg-purple-300
hover:text-purple-800 rounded transition duration-300 ">Log out

{%endblock%

{% block mdNav%

<a href="{ {url_for('profile') }}" class="block py-2 px-4 text-sm hover:bg-gray-
200">Profile

Log
out

{%endblock%

{% block searchJob%

<!-- text Typing Animation-->

<div class="py-5 flex justify-center text-3xl px-4">

<p>Inspiring Jobs for the future

!</p>

</div>

<!-- form to search job -->

<div class="py-5 px-4 ">

<form action="/search" class="md:flex md:justify-center" id="form-1"
method="POST">

<div class="">

<input class="p-2 my-2 w-full border-[1px]" type="text" placeholder="Role: Fullstack"
name="role">

</div>

<div>

<input class="p-2 my-2 w-full border-[1px]" type="text" placeholder="City:
Chennai" name="city">

</div>

<div>

<input class="p-2 my-2 w-full border-[1px]" type="text" placeholder="Country:India"
name="country">

```

</div>

<div class="md:flex ">
  {% if session['user']['jobFetchCount'] == 3 %}
    <button class="p-2 my-2 w-full border-[2px] bg-purple-600 rounded text-white hover:bg-white hover:text-purple-800 hover:border-purple-400 " id="searchBtn">
      Go
    </button>
  {% else %}
    <button id="goButton" class="p-2 my-2 w-full border-[2px] bg-purple-600 rounded text-white hover:bg-white hover:text-purple-800 hover:border-purple-400 " >
      Go
    </button>
  {% endif %}
  <div
    class="relative p-2 my-2 w-full border-[2px] bg-green-400 rounded text-white
    hover:bg-white hover:text-green-800 hover:border-green-400 text-center ">
    <a class="block z-1 after:absolute after:top-0 after:right-0 after:left-0
    after:bottom-0 after:content-[' ']"
      href="{ { url_for('history',history=0) } }">History</a>
  </div>
</div>

<!-- this is prompt when user clicks search job button -->
<!-- It's hidden only visible when user click 'GO' or Job Search button -->
<div>
  <div id="popup-modal" tabindex="-1"
    class="hidden overflow-y-auto overflow-x-hidden fixed top-0 right-0 left-0 z-50
    md:inset-0 h-modal md:h-full justify-center items-center"
    aria-hidden="true">
    <div class="flex justify-center items-center h-screen">
      <div class="relative p-4 w-full max-w-md h-full md:h-auto">
        <div class="relative bg-white rounded-lg shadow dark:bg-gray-700">
          <button type="button" id="promptClose"
            class="absolute top-3 right-2.5 text-gray-400 bg-transparent
            hover:bg-gray-200 hover:text-gray-900 rounded-lg text-sm p-1.5 ml-auto inline-flex items-center
            dark:hover:bg-gray-800 dark:hover:text-white"

```

```

data-modal-toggle="popup-modal">
    <svg aria-hidden="true" class="w-5 h-5" fill="currentColor"
viewBox="0 0 20 20"
    xmlns="http://www.w3.org/2000/svg">
    <path fill-rule="evenodd"
        d="M4.293 4.293a1 1 0 011.414 0L10 8.586l4.293-4.293a1 1 0
111.414 1.414L11.414 10l4.293 4.293a1 1 0 01-1.414 1.414L10 11.414l-4.293 4.293a1
1 0 01-1.414-1.414L8.586 10 4.293 5.707a1 1 0 010-1.414z"
        clip-rule="evenodd"></path>
    </svg>
    <span class="sr-only">Close modal</span>
</button>
<div class="p-6 text-center">
    <svg aria-hidden="true" class="mx-auto mb-4 w-14 h-14 text-gray-400
dark:text-gray-200"
        fill="none" stroke="currentColor" viewBox="0 0 24 24"
    xmlns="http://www.w3.org/2000/svg">
        <path stroke-linecap="round" stroke-linejoin="round" stroke-
width="2"
            d="M12 8v4m0 4h.01M21 12a9 9 0 11-18 0 9 9 0 0118
0z"></path>
    </svg>
    <div class="mb-5 text-lg font-normal text-gray-500 dark:text-gray-
400">
        <h3>In Free tier you are allowed only 3 Search </h3>
        <h3>Are you sure use one</h3>
    </div>
    <button value="submit" type="button" id="confirmSearch"
form="form-1" type="submit" class="text-white bg-green-600 hover:bg-green-800 focus:ring-4
focus:outline-none focus:ring-green-300 dark:focus:ring-green-800 font- medium rounded-lg text-sm
inline-flex items-center px-5 py-2.5 text-center mr-2">
        submit </button>
    <button data-modal-toggle="popup-modal" type="button"
id="cancelBtn"
        class="text-gray-500 bg-white hover:bg-gray-100 focus:ring-4
focus:outline-none focus:ring-gray-200 rounded-lg border border-gray-200 text-sm font-medium
px-5 py-2.5 hover:text-gray-900 focus:z-10 dark:bg-gray-700 dark:text-gray-300

```

```

dark:border-gray-500 dark:hover:text-white dark:hover:bg-gray-600 dark:focus:ring-gray-600">No,
        cancel</button>
    </div>
</div>
</div>
</div>
</div>
</div>
<!-- end -->

</form>

<!-- this is alert when user use more than 3 searches -->
<!-- It's hidden only visible when user click 'GO' or Job Serach button -->
<div class="md:flex md:justify-center">
    <div id="alertBorder" class="hidden flex p-4 mb-4 bg-yellow-100 border-t-4border-
yellow-500 dark:bg-yellow-200" role="alert">
        <svg class="flex-shrink-0 w-5 h-5 text-yellow-700" fill="currentColor"
viewBox="0 0 20 20" xmlns="http://www.w3.org/2000/svg"><path fill-rule="evenodd"
d="M18 10a8 8 0 11-16 0 8 8 0 0116 0zm-7-4a1 1 0 11-2 0 1 1 0 012 0zM9 9a1 1 0 00
2v3a1 1 0 001 1h1a1 1 0 100-2v-3a1 1 0 00-1-1H9z" clip-rule="evenodd"></path></svg>
        <div class="ml-3 text-sm font-medium text-yellow-700">
            You Used 3 searches If you need more Please Mail to this ID <a href="#"
class="font-semibold
                                underline
                                hover:text-yellow-
800">prakash.ece19@gmail.com"</a>.
        </div>
        <button type="button" id="alertBorderClose" class="ml-auto -mx-1.5 -my-1.5 bg-
yellow-100 dark:bg-yellow-200 text-yellow-500 rounded-lg focus:ring-2 focus:ring-
yellow-400 p-1.5
hover:bg-yellow-200 dark:hover:bg-yellow-300 inline-flex h-8 w-8" data-
dismiss-target="#alert-
border-4" aria-label="Close">
            <span class="sr-only">Dismiss</span>
            <svg aria-hidden="true" class="w-5 h-5" fill="currentColor" viewBox="0 0 20 20"
xmlns="http://www.w3.org/2000/svg"><path fill-rule="evenodd" d="M4.293 4.293a1 1 0 011.414
0L10 8.586l4.293-4.293a1 1 0 011.414 1.414L11.414 10l4.293 4.293a1 1 0
01-1.414 1.414L10 11.414l-4.293 4.293a1 1 0 01-1.414-1.414L8.586 10 4.293 5.707a1 1

```

```

0 010-1.414z" clip-rule="evenodd"></path></svg>
    </button>
  </div>
</div>
<!-- end -->

<!-- history tab buttons -->
<div class="flex justify-center py-3">
  <div class="bg-white px-3 mx-2 border-2 border-purple-300 hover:scale-110
relative ">
    <a class="block z-1 after:absolute after:top-0 after:right-0 after:left-0
after:bottom-0 after:content-[ ' ]" href="{ {url_for('history',history=0)} }">1</a>
  </div>
  <div class="bg-white px-3 mx-2 border-2 border-purple-300 hover:scale-110
relative">
    <a class="block z-1 after:absolute after:top-0 after:right-0 after:left-0
after:bottom-0 after:content-[ ' ]" href="{ {url_for('history',history=1)} }">2</a>
  </div>
  <div class="bg-white px-3 mx-2 border-2 border-purple-300 hover:scale-110
relative">
    <a class="block z-1 after:absolute after:top-0 after:right-0 after:left-0
after:bottom-0 after:content-[ ' ]" href="{ {url_for('history',history=2)} }">3</a>
  </div>
</div>

<!-- error msg -->
<div class=" px-4 md:px-56 " id="historyError">
  { % if error % }
  <div id="alert-4" class="flex justify-center p-4 mb-4 bg-yellow-100 rounded-lg
dark:bg-yellow-200" role="alert">
    <svg aria-hidden="true" class="flex-shrink-0 w-5 h-5 text-yellow-700
dark:text-yellow-800" fill="currentColor" viewBox="0 0 20 20"
xmlns="http://www.w3.org/2000/svg"><path fill-rule="evenodd" d="M18 10a8 8 0 11-16
0 8 8 0 0116 0zm-7-4a1 1 0 11-2 0 1 1 0 012 0zM9 9a1 1 0 00 2v3a1 1 0 01 1h1a1 1 0
100-2v-3a1 1 0 00-1-1H9z" clip-rule="evenodd"></path></svg>
    <span class="sr-only">Info</span>
    <div class="ml-3 text-sm font-medium text-yellow-700 dark:text-yellow-

```

800">

```
        {{error}} </div>
        <button id="historyErrorBtn" type="button" class="ml-auto -mx-1.5 -my-
1.5 bg-yellow-100 text-yellow-500 rounded-lg focus:ring-2 focus:ring-yellow-400 p-1.5 hover:bg-
yellow-200 inline-flex h-8 w-8 dark:bg-yellow-200 dark:text-yellow-600 dark:hover:bg-yellow-
300" data-dismiss-target="#alert-4" aria-label="Close">
            <span class="sr-only">Close</span>
            <svg aria-hidden="true" class="w-5 h-5" fill="currentColor" viewBox="0 0
20 20" xmlns="http://www.w3.org/2000/svg"><path fill-rule="evenodd" d="M4.293 4.293a1 1 0
011.414 0L10 8.586l4.293-4.293a1 1 0 111.414 1.414L11.414 10l4.293 4.293a1 1 0 01-1.414
1.414L10 11.414l-4.293 4.293a1 1 0 01-1.414-1.414L8.586 10 4.293 5.707a1 1 0 010-1.414z"
clip-rule="evenodd"></path></svg>
        </button>
    </div>
    {% endif %}
</div>
<!-- end error -->

</div>
<!-- form to search job end -->
{%endblock%}

{% block body %}
<!-- <h2 class="text-center p-10 my-14 mx-28 rounded-xl bg-blue-500 text-white text-
5xl">heloo {{username}} <br> Job is Loading soon</h2> -->
<!-- card start -->
{% for job in jobs %}

    <div class="justify-around flex mt-10 ">
        <button class="w-11/12 md:w-10/12 bg-white rounded-xl shadow-2xl hover:scale-105 jobid
relative " data-id="{{job['id']}}" >
            <div class="flex justify-between p-5 md:p-10 lg:p-14 text-gray-800 text-left">
                <div class="w-8/12">
                    <h2 class="w-40 md:w-96 lg:w-full text-purple-600 font-semibold py-1 text-xlmd:text-
2xl">{{job["jobTitle"]}}</h2>
                    <h2 class="w-40 md:w-96 py-1 text-lg md:text-
xl">{{job['companyName']}}</h2>
```

```

        {% if job['companyUrl'] !=None %}
            <a class="w-40 text-gray-500 py-1 md:text-lg " href="{ {job['companyUrl'] }}"
target="_blank">{ {job['companyUrl'] }}</a>
        {%endif%}
    </div>
    <div>
        
    </div>
</div>
<div class="text-left flex md:text-lg lg:px-14 justify-between pb-5 px-5 md:px-10md:pb-
10 text-gray-400">
    <div class="">
        <h4>Location:</h4>
        <h4>{ {job['location'] }}</h4>
        {% if count == None %}
            <h4><a href="{ {url_for('apply',x=job['cloudId']) }}" class="block z-1
after:absolute after:top-0 after:right-0 after:left-0 after:bottom-0 after:content-[' ']">Click here for
<span class="text-purple-500">more details</span></a></h4>
        {% else %}
            <h4><a href="{ {url_for('applys',x=job['cloudId'],count=count) }}" class="block z-
1 after:absolute after:top-0 after:right-0 after:left-0 after:bottom-0 after:content-[' ']">Click here
for <span class="text-purple-500">more details</span></a></h4>
        {%endif%}
    </div>
    <div>
        <h4>Salary:</h4>
        {% if job['salary'] == None %}
            <h4>Not Disclosed</h4>
        {% else %}
            <h4>{ {job['salary'] }}</h4>
        {%endif%}
    </div>
</div>
</button>
</div>

```

```
{% endfor %}  
<!-- card end -->
```

```
{% block js %}  
<script src="{{url_for('static', filename='job.js')}}"></script>  
<script src="{{url_for('static', filename='history.js')}}"></script>  
<script src="{{url_for('static', filename='searchConfirm.js')}}"></script>  
{%endblock%}  
  
{% endblock%}
```

githublink:

[https:// github.com/IBM-EPBL/IBM-Project-48675-1660811187](https://github.com/IBM-EPBL/IBM-Project-48675-1660811187)