

Final report

Skill and job recommender application

Mentors:

- 1) Krishna Chaitanya**
- 2) Dr A.R.Kavitha**

Team ID : PNT2022TMID24872

Team members:

- 1) Karthik P (Team leader)**
- 2) Nandakumar R**
- 3) Aswin kumar P**
- 4) Jasper kirubakaran J**

<u>INDEX:</u>	Pg:
1. INTRODUCTION	
1.1 Project Overview	5
1.2 Purpose	
2. LITERATURE SURVEY	6
2.1 Existing problem	
2.2 References	7
2.3 Problem Statement Definition	
3. IDEATION & PROPOSED SOLUTION	8
3.1 Empathy Map Canvas	
3.2 Ideation & Brainstorming	9
3.3 Problem statement	10
3.4 Problem solution fit	11
3.5 Proposed Solution	12
4. REQUIREMENT ANALYSIS	14
4.1 Functional requirement	
4.2 Non-Functional requirements	15
5. PROJECT DESIGN	
5.1 Data Flow Diagrams	

5.2	Solution Architecture	16
5.3	User Stories	17
6.	PROJECT PLANNING & SCHEDULING	18
6.1	Sprint Planning & Estimation	
6.2	Sprint Delivery Schedule	19
6.3	Jira report	20
7.	CODING & SOLUTIONING	22
7.1	Registration and login	
7.2	Apply job	31
7.3	View job	36
7.4	Chatbot	39
7.5	Send grid	
8.	TESTING	45
8.1	Test Cases	
8.2	User Acceptance Testing	46
9.	RESULTS	47
9.1	Performance Metrics	
10.	ADVANTAGES & DISADVANTAGES	48

11.	CONCLUSION	48
12.	FUTURE SCOPE	49
13.	APPENDIX	49
13.1	Source Code	
13.2	GitHub & Project Demo Link	71

Skill and Job recommender application:

1.Introduction:

Now a days Searching a job through the internet is a common task using various platforms such as linked in, websites and other social media applications such as Instagram. Basically a job seeker can search for a job either by searching for job vacancies that suits his/her profile or by creating a profile stating his/her educational details, professional skill and personal experiences and receive job details based on the provided data. Job recommender systems have implemented many type of recommenders such as content-based filtering, collaborative filtering, knowledge based and hybrid approaches.

1.1 Overview:

Skill and job recommender:

Now a days its very difficult for people to find a job in today's competitive world, but skill and job recommender application is used to recommend various types of available vacant job roles to the job seekers. Additionally, it describes the various career paths open to professionals in all fields as well as the various business models used by those employers. Our application recommends job based on the skillset provided by the user.

1.2 Purpose:

- Better user experience
- Higher engagement
- Skill improvement

- Revenue opportunities
- Job opportunities
- Handling high user applications

2. Literature survey:

Abstract:

Skill / Job Recommender Application 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 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.

2.1 Existing system:

1. Based on their skill set, students and job seekers find their ideal position.
2. Including Intelligent CHATBOT in applications for employment recommendations.
3. An analysis of LinkedIn as a tool for recruiters and job seekers.
4. Online file sharing and storage

2.2 References:

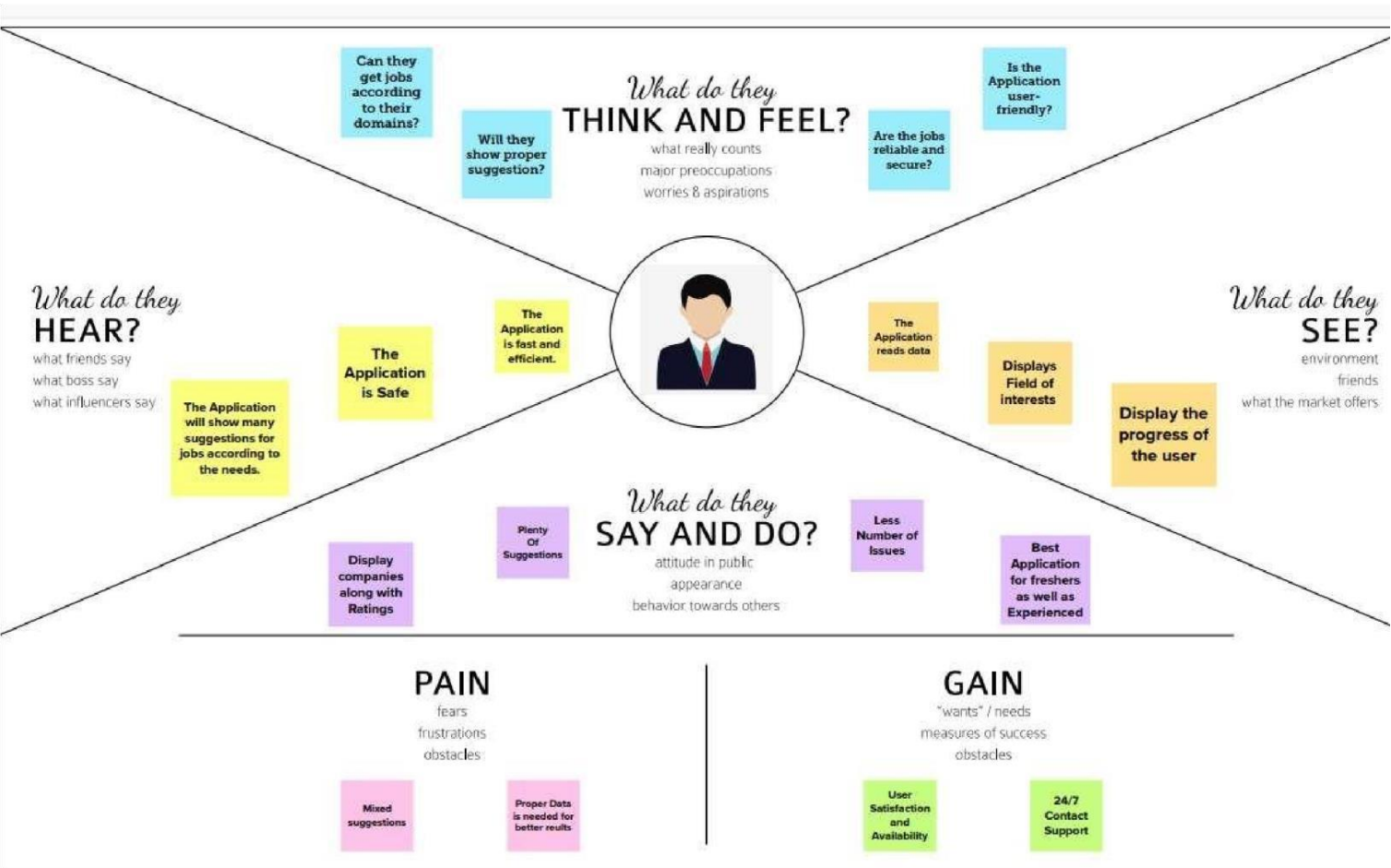
- * 2019, IEEE 6th international conference in cloud computing and intelligence system.
- * 2020, IEEE conference in big data.
- * International journal for research in applied science and engineering technology.
- * The international conference in artificial intelligence and smart city applications.
- *Students computer science and engineering, kk Wagh college of engineering Nashik India.

2.3 Problem statement definition:

A job seeker always spends hours searching through the massive amount of recruiting information on the internet to identify ones that are helpful. People who don't have any industry experience frequently don't know exactly what they need to learn in order to get a career that's right for them. We deal with the issue of suggesting suitable occupations to those looking for new employment. The goal of job recommender technology is to assist job seekers in Locating the positions that fit their skill set.

3. Ideation and proposed solution

3.1 Empathy map:



3.2 Brainstorming:

Brainstorm & idea prioritization

Use the template in your own brainstorming sessions or your team will generate ideas, map them out, and start shaping concepts over if you're not doing it like some steps.

1. Problem space
2. Theme selection
3. Brainstorming

0 Define your challenge

Write down the problem you're trying to solve. It should be a single sentence. It should be a problem you're trying to solve. It should be a problem you're trying to solve.

1. Problem

2. Problem

3. Problem

0 Define your problem statement

Write down the problem you're trying to solve. It should be a single sentence. It should be a problem you're trying to solve. It should be a problem you're trying to solve.

1. Problem

2. Problem

3. Problem

0 Brainstorm

Write down the problem you're trying to solve. It should be a single sentence. It should be a problem you're trying to solve. It should be a problem you're trying to solve.

1. Problem

2. Problem

3. Problem

0 Group ideas

Write down the problem you're trying to solve. It should be a single sentence. It should be a problem you're trying to solve. It should be a problem you're trying to solve.

1. Problem

2. Problem

3. Problem

0 Prioritize

Write down the problem you're trying to solve. It should be a single sentence. It should be a problem you're trying to solve. It should be a problem you're trying to solve.

1. Problem

2. Problem

3. Problem

0 After you brainstorm

Write down the problem you're trying to solve. It should be a single sentence. It should be a problem you're trying to solve. It should be a problem you're trying to solve.

1. Problem

2. Problem

3. Problem

3.3 Problem statement:

Problem Statement I:



Problem Statement II:



3.4 Problem solution fit:

1. CUSTOMER SEGMENT(S) CS

Who is your customer?

- The Job Seekers are the ones who are in need of jobs.
- The employers who provide jobs using our application/software.

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

- Lack of Knowledge and skills.
- No cash and less number of available devices
- Inconvenient Access to support.
- Improper Device Configuration.

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital note taking.

- Ease of Access and Security.
- Plenty of Suggestions.
- Chances of negative recommendations.
- Huge amount of data is needed.

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

- Lack of data analytics capability.
- Inability to capture changes and updates in customer behaviour.
- Low Adaptability.
- Less Reliability.

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists? What is the back story behind the need to do this job? i.e. customers have to do it because of the change in regulations.

- Due To Evolving and rapidly growing technologies, there is a need for the customers to learn everything.
- To overcome Unemployment and increasing employment rate.

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done? i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

- Using Standardized Computing Techniques and boosting algorithms.
- Data and skillset is needed for proper suggestions.

3. TRIGGERS TR

What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news

- Compete among customers will make them triggered.
- Social media plays a major role for competitions.

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards? i.e. lost, insecure > confident, in control - use it in your communication strategy & design

- Before : Customer feels that he/she is hopeless, frustrated, less concentration and negative thinking.
- After : Once they get suggestions from our application, they will be happy and they will feel that they have accomplished their goals.

10. YOUR SOLUTION SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality. If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

The proposed system consists of the following three major modules, which are completed as part of this research as follows:

- Data collection and preprocessing followed by the unification of the database.
- Recommendation of suitable results using a hybrid system of content-based and collaborative filtering.
- Development of a fully functional user interface in the form of a web application.

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE

What kind of actions do customers take online? Extract online channels from #7

The software/application is fully functional in online mode. It requires stable internet connection for login, data collection, evaluation and job suggestions.

8.2 OFFLINE

What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

When there is a need for customer support or in-person interview, customer should must be there.

3.4 Proposed solution:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	<p>*Having a variety of abilities but unsure of which position would be best for you? No need to be concerned! With our skill recommender solution, either a skilled or a fresher user may sign up, search for jobs using the search bar, or speak with the chatbot directly to land their ideal position.</p> <p>*To create a complete web application that can show available jobs based on the user's skill set. The database contains both user and their data. In addition to that, company can also add their available job offers to get employees for a particular role.</p>
2.	Idea / Solution description	<p>The implemented ideas are:</p> <ul style="list-style-type: none">*IBM DB2 is used for storing the data.*Chatbot is implemented by Watson Assistant.*Job search API is used for searching jobs based on the educational qualification and skills. *In addition to that Kubernetes cluster and container registry are used.*With the help of the above tools and techniques skill and job recommender application will be implemented and developed.

3.	Novelty / Uniqueness	<p>*The Skill & Job Recommender System is capable of offering suggestions based on users data and preferences. These systems processes information using information filtering techniques and present them a wide range of opportunities.</p> <p>*The web applications provides recommendations based on the user's educational qualifications and skills.</p> <p>*It doesn't recommend jobs that do not suit the user's profile.</p>
4.	Social Impact / Customer Satisfaction	<p>Impacts include:</p> <p>*Job seekers can easily get jobs.</p> <p>* Recruitment of employees gets easier for employers.</p> <p>*It's time saving for both the company and the user.</p> <p>*Companies can provide vacant job roles and find ones for that specific role.</p>
5.	Business Model (Revenue Model)	<p>Skill/Job Recommender system is needed because:</p> <p>*Each organisation needs to recruit employees for a particular field/department.</p> <p>*For instance, in educational sector, teachers/professors get hired based on their educational background.</p> <p>* Skill and job Recommender application is used for the hiring process.</p> <p>* The usage of skill and job Recommender application is widely used for the recruitment process.</p>

6.	Scalability of the Solution	<p>*IBM DB2 is used by many businesses, which offers a data platform for both transactional and analytical operations. It ensures that data is continuously available such that transactional process and analytics function smoothly. *Watson Assistant is used to automate communications with end customers.</p> <p>* Jobs can be recommended.</p> <p>* Kubernetes cluster helps application to run on multiple machines and environment.</p> <p>*Container Registry manages all the user details in single place.</p>
----	-----------------------------	---

4.Requirement analysis:

4.1 Functional requirements:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Register Form Registration through Gmail Registration through LinkedIn
FR-2	User Confirmation	Confirmation via Email
FR-3	User Browser	Browse For Job
FR-4	User post	Post Job Vacancies to Seeker
FR-5	User Learn	User can gain skills using this platform.
FR-6	Recommendation	Recommended as per search

4.2 Non-functional requirements:

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	The UI for this project is user-friendly and asynchronous loadable application.
NFR-2	Security	The application is secured and authentication is provided.
NFR-3	Reliability	The system/application must perform without fail for atleast 95% of the time.
NFR-4	Performance	The application can handle many processes at a time.
NFR-5	Availability	The application should perform without any failures.
NFR-6	Scalability	Scalability is good.

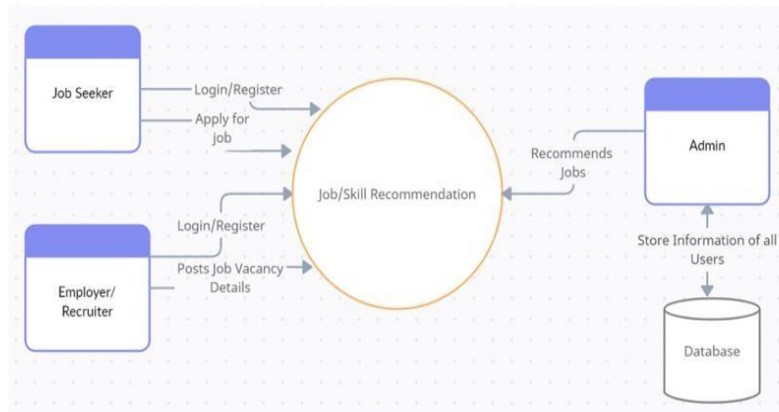
5. Project design:

5.1 Data flow diagram:

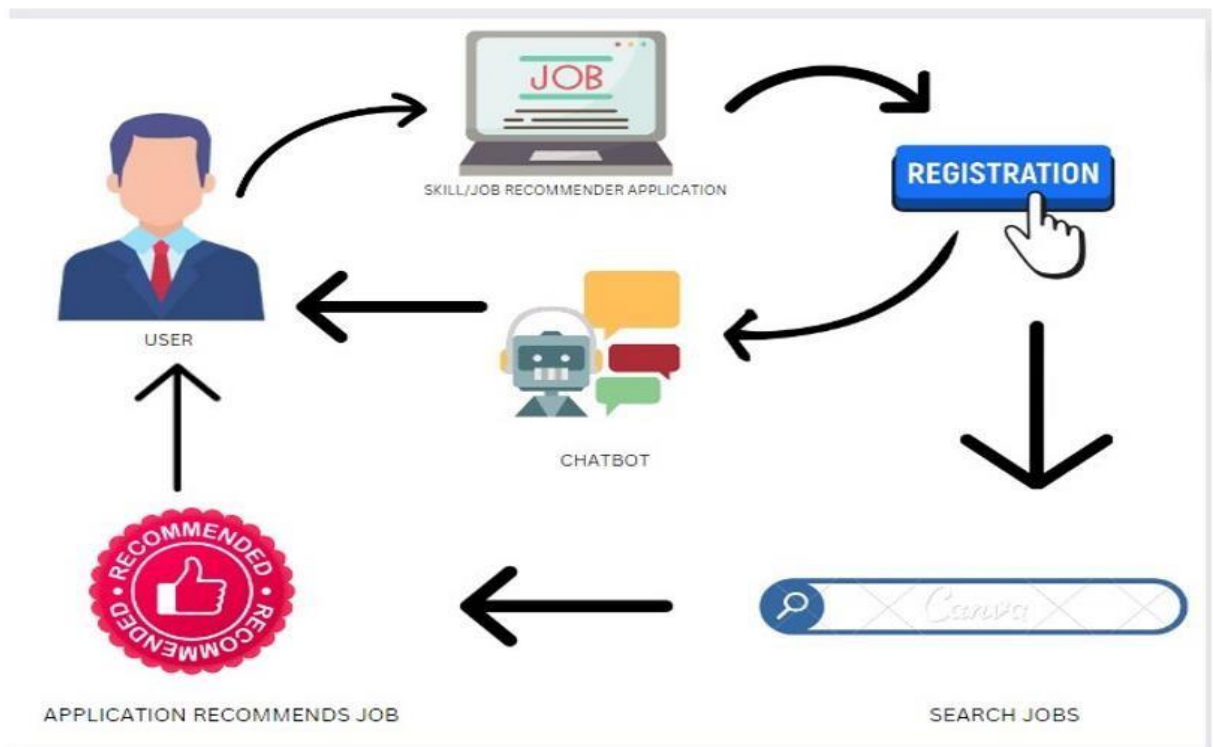
Data Flow Diagrams:

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.

Example:



5.2 Solution architecture:



5.3 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	Registration	USN-1	As a user, I can register for the application 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 <u>Linkedin</u>	I can register & access the dashboard with <u>Linkedin</u> 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 log in and view my account.	High	Sprint-1
	Dashboard	USN-6	As a user, I can view the dashboard with my profile and Job button where I can see a lot of job recommendations.	I can access my dashboard and view the Recommendations.	High	Sprint-1
Customer (Job Seeker)	Upload	USN-7	As a job seeker, I can upload my resume and skills.	I can view my resume.	High	Sprint-2
<u>Customer</u> (Employer/ Recruiter)	Post Jobs	USN-8	As a Recruiter, I can post job vacancies in the application.	I can view the job posts.	Medium	Sprint-1
Administrator	Recommendation	USN-9	As an administrator, I can recommend jobs.	I can give job recommendations.	High	Sprint-3
	Maintenance	USN-10	As an administrator, I can access the user details stored in database.	I can access the database.	High	Sprint-4

6.Project planning and scheduling:

6.1Sprint planning and estimation:

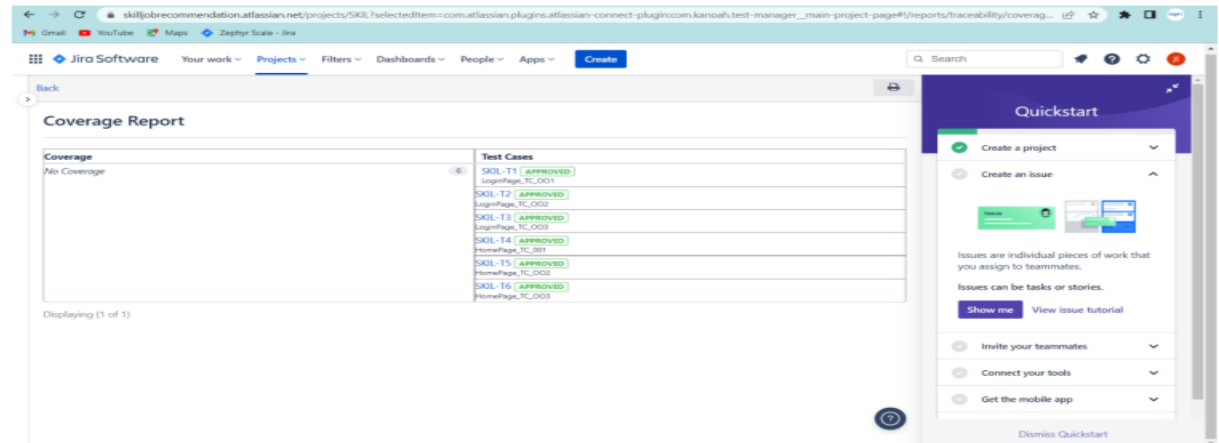
Sprint	Functional requirement	User story number	User story task	Story points	Priority	Team members
Sprint 1	Registration	USN-1	As a user I can register for the application by entering my email, password and confirming my password.	2	High	Karthik, Nandakumar Aswinkumar Jasper kirubakaran
Sprint 3	Access	USN-2	As a user I can access through cloud.	1	Low	Karthik, Nandakumar Aswinkumar Jasper kirubakaran
Sprint 1	Login	USN-3	As a user I can log into the application using my email and password.	1	Low	Karthik, Nandakumar Aswinkumar Jasper kirubakaran
Sprint 1	Job search	USN-4	As a user I can view for job.	2	High	Karthik, Nandakumar Aswinkumar Jasper kirubakaran
Sprint 1	Support	USN-5	As a user I can get some		High	Karthik, Nandakumar Aswinkumar

6.2 Sprint delivery schedule:

Sprint	Total story points	Duration	Sprint start date	Sprint end date	Story points completed (as on planned end date)	Sprint release date (Actual)
Sprint 1	20	5	24 Oct 2022	28 Oct 2022	20	28 Oct 2022
Sprint 2	20	5	31 Oct 2022	4 Nov 2022	19	4 Nov 2022
Sprint 3	20	5	5 Nov 2022	9 Nov 2022	19	9 Nov 2022
Sprint 4	20	5	15 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 Jira report:

Coverage Report



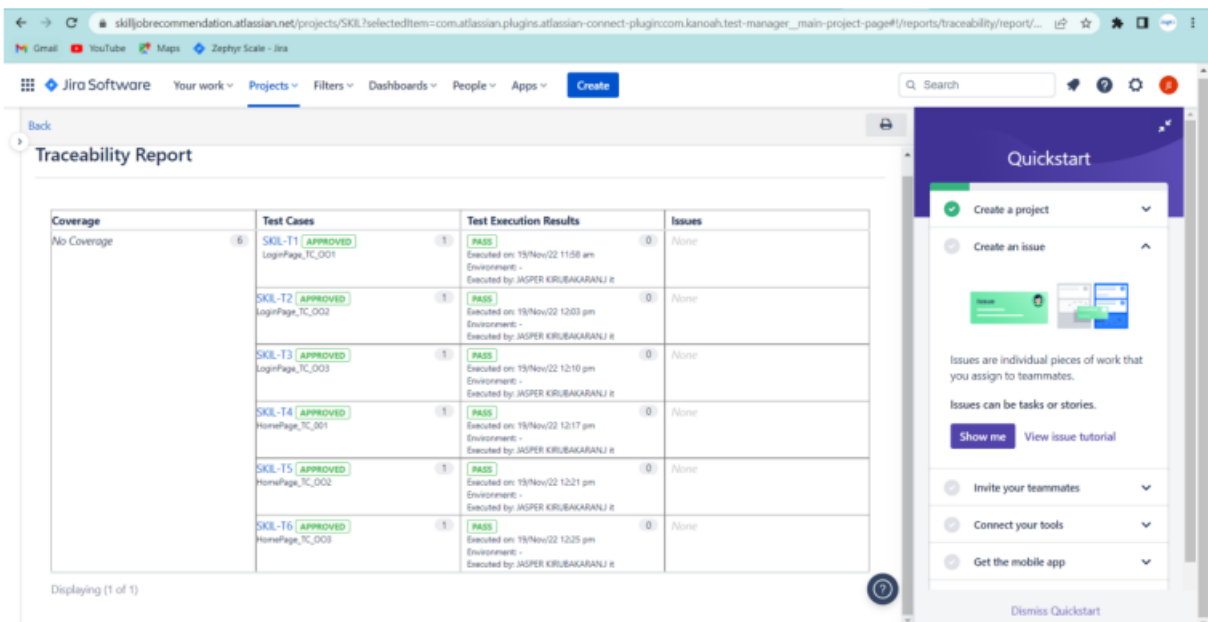
The screenshot shows the Jira Coverage Report interface. On the left, under the 'Coverage' tab, it displays 'No Coverage'. On the right, under the 'Test Cases' tab, a list of six test cases is shown, all marked as 'APPROVED'.

Test Cases
SKIL-T1 (APPROVED) LoginPage_TC_001
SKIL-T2 (APPROVED) LoginPage_TC_002
SKIL-T3 (APPROVED) LoginPage_TC_003
SKIL-T4 (APPROVED) HomePage_TC_001
SKIL-T5 (APPROVED) HomePage_TC_002
SKIL-T6 (APPROVED) HomePage_TC_003

Displaying (1 of 1)

On the right side of the interface, there is a 'Quickstart' sidebar with options like 'Create a project', 'Create an issue', 'Invite your teammates', 'Connect your tools', and 'Get the mobile app'.

Traceability Report



The screenshot shows the Jira Traceability Report interface. It displays a table with four columns: Coverage, Test Cases, Test Execution Results, and Issues.

Coverage	Test Cases	Test Execution Results	Issues
No Coverage	SKIL-T1 (APPROVED) LoginPage_TC_001	PASS Executed on: 19/Nov/22 11:58 am Environment: - Executed by: JASPER KRUBAKARANJ R	None
	SKIL-T2 (APPROVED) LoginPage_TC_002	PASS Executed on: 19/Nov/22 12:03 pm Environment: - Executed by: JASPER KRUBAKARANJ R	None
	SKIL-T3 (APPROVED) LoginPage_TC_003	PASS Executed on: 19/Nov/22 12:10 pm Environment: - Executed by: JASPER KRUBAKARANJ R	None
	SKIL-T4 (APPROVED) HomePage_TC_001	PASS Executed on: 19/Nov/22 12:17 pm Environment: - Executed by: JASPER KRUBAKARANJ R	None
	SKIL-T5 (APPROVED) HomePage_TC_002	PASS Executed on: 19/Nov/22 12:21 pm Environment: - Executed by: JASPER KRUBAKARANJ R	None
	SKIL-T6 (APPROVED) HomePage_TC_003	PASS Executed on: 19/Nov/22 12:25 pm Environment: - Executed by: JASPER KRUBAKARANJ R	None

Displaying (1 of 1)

On the right side of the interface, there is a 'Quickstart' sidebar with options like 'Create a project', 'Create an issue', 'Invite your teammates', 'Connect your tools', and 'Get the mobile app'.

Traceability matrix

The screenshot shows the Jira Software interface with the 'Traceability matrix' view selected. The main content area displays a table with columns for 'Test Case' and 'Coverage'. The 'Test Case' column lists SKIL-T1, SKIL-T2, SKIL-T3, SKIL-T4, SKIL-T5, and SKIL-T6. The 'Coverage' column shows 'No Coverage' for all test cases. A green bar at the bottom indicates 'Last test execution: Pass'. A 'Quickstart' sidebar is visible on the right with options like 'Create a project', 'Create an issue', 'Invite your teammates', 'Connect your tools', and 'Get the mobile app'.

Test Case	Coverage
SKIL-T1 - LoginPg...	No Coverage
SKIL-T2 - LoginPg...	No Coverage
SKIL-T3 - LoginPg...	No Coverage
SKIL-T4 - HomePg...	No Coverage
SKIL-T5 - HomePg...	No Coverage
SKIL-T6 - HomePg...	No Coverage

Displaying (1 of 1)

Last test execution: Pass

Traceability Tree

The screenshot shows the Jira Software interface with the 'Traceability Tree' view selected. The main content area displays a table with columns for 'Traceability' and 'Summary'. The 'Traceability' column lists test cases SKIL-T1 through SKIL-T6. The 'Summary' column shows the test case name, the date and time of execution, and the status (PASS). A 'Quickstart' sidebar is visible on the right with options like 'Create a project', 'Create an issue', 'Invite your teammates', 'Connect your tools', and 'Get the mobile app'.

Traceability	Summary
No Coverage	
Covered by Test Case SKIL-T1	LoginPage_TC_001
Executed on 19/Nov/22 11:58 am	PASS Executed by JASPER KIRUBAKARAN.J it
Covered by Test Case SKIL-T2	LoginPage_TC_002
Executed on 19/Nov/22 12:03 pm	PASS Executed by JASPER KIRUBAKARAN.J it
Covered by Test Case SKIL-T3	LoginPage_TC_003
Executed on 19/Nov/22 12:10 pm	PASS Executed by JASPER KIRUBAKARAN.J it
Covered by Test Case SKIL-T4	HomePage_TC_001
Executed on 19/Nov/22 12:17 pm	PASS Executed by JASPER KIRUBAKARAN.J it
Covered by Test Case SKIL-T5	HomePage_TC_002
Executed on 19/Nov/22 12:21 pm	PASS Executed by JASPER KIRUBAKARAN.J it
Covered by Test Case SKIL-T6	HomePage_TC_003
Executed on 19/Nov/22 12:25 pm	PASS Executed by JASPER KIRUBAKARAN.J it

7.Coding and solutioning:

7.1 Registration and login:

```
<!DOC
TYPE
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>Registration and Login</title>
        <link rel="stylesheet" href="./style.css"/>
        <script defer
src="https://use.fontawesome.com/releases/v5.15.4/js/all.js"
integrity="sha384-
rOA1PnstxnOBLzCLMcre8ybwbTmemjzdNiILg8O7z1lUkLX
ozs4DHonlDtnE7fpc" crossorigin="anonymous"></script>
        <style>

        @import
url('https://fonts.googleapis.com/css?family=Montserrat:400,80
0');

        * {
            box-sizing: border-box;
        }
```

```

    body {
        background: #f6f5f7;
        display: flex;
        justify-content: center;          align-
items: center;        flex-direction: column;
        font-family: 'Montserrat', sans-
serif;
        height: 100vh;
        margin: -20px 0 50px;
    }

    h1 {
        font-weight: bold;
        margin: 0;
    }

    h2 {
        text-align: center;
    }

    p {
        font-size: 14px;
        font-weight: 100;        line-
height: 20px;        letter-spacing:
0.5px;        margin: 20px 0 30px;
    }

    span {
        font-size: 12px;
    }

    a {

```

```

        color: #333;
font-size: 14px;
text-decoration: none;
        margin: 15px 0;
    }

```

```

        button {
            border-radius: 20px;
border: 1px solid #41d3ff;
background-color: #41d3ff;
color: #FFFFFF;            font-size:
12px;            font-weight: bold;
            padding: 12px 45px;            letter-
spacing: 1px;            text-transform:
uppercase;            transition: transform
80ms ease-in;
        }

                button:active {            transform: scale(0.95);
    }

            button:focus {            outline: none;
    }

            button.ghost {            background-color: transparent;
border-color: #FFFFFF;
    }

```

```

        form {
            background-color: #FFFFFF;
display: flex;
            align-items: center;
            justify-content:
center;            flex-
direction: column;

```



```

        padding: 0 50px;
        height: 100%;
        text-align: center;
    }

    input {
        background-color: #eee;
        border: none;
        padding: 12px 15px;
        margin: 8px 0;
        width: 100%;
    }

    .container {
        background-color: #fff;
        border-radius: 10px;
        box-shadow: 0 14px 28px rgba(0,0,0,0.25),
                    0 10px 10px rgba(0,0,0,0.22);
        position: relative;
        overflow: hidden;
        width: 768px;           max-
width: 100%;           min-height:
480px;
    }

    .form-container {
        position: absolute;
        top: 0;
        height: 100%;
        transition: all 0.6s ease-in-out;
    }

    .sign-in-container {
        left: 0;
        width: 50%;

```

```

        z-index: 2;
    }

    .container.right-panel-active .sign-in-container {
        transform: translateX(100%);
    }

    .sign-up-container
    {
        left: 0;
        width: 50%;
        opacity: 0;
        z-index: 1;
    }

    .container.right-panel-active .sign-up-container
    {
        transform: translateX(100%);
        opacity: 1;
        z-index: 5;
        animation: show 0.6s;
    }

    @keyframes show {
        0%, 49.99% {
            opacity: 0;
            z-index: 1;
        }

        50%, 100% {
            opacity: 1;
            z-index: 5;
        }
    }

```

```

.overlay-container { position:
absolute;
    top: 0; left:
    50%;
    width:
    50%;
    height:
    100%;
    overflow:
    hidden;
    transition: transform 0.6s ease-in-out;
    z-index: 100;
}

.container.right-panel-active .overlay-container{
    transform: translateX(-100%);
}

```

```

.overlay {
background: #41d3ff;
    background: -webkit-linear-gradient(to right, #41d3ff,
    #FF416C);
    background: linear-gradient(to right, #41d3ff, #FF416C);
    background-repeat: no-repeat;
    background-size: cover;
background-position: 0 0;
color: #FFFFFF;
    position: relative;
    left: -100%;
    height: 100%;
    width: 200%;
    transform: translateX(0); transition:
transform 0.6s ease-in-out;
}

```

```
.container.right-panel-active .overlay {  
transform: translateX(50%);  
}
```

```
.overlay-panel {  
position: absolute;  
    display: flex; align-  
    items: center;  
    justify-content:  
    center; flex-  
    direction: column;  
    padding: 0 40px;  
    text-align: center;  
    top: 0;  
    height: 100%;  
    width: 50%;  
    transform: translateX(0); transition:  
    transform 0.6s ease-in-out;  
}
```

```
.overlay-left {  
    transform: translateX(-20%);  
}
```

```
.container.right-panel-active .overlay-left {  
transform: translateX(0);  
}
```

```
.overlay-right {  
    right: 0;  
    transform: translateX(0);  
}
```

```

.container.right-panel-active .overlay-right {
  transform: translateX(20%);
}

.social-container {
  margin: 20px 0;
}

.social-container a { border: 1px solid
#DDDDDD;
border-radius:
50%; display:
inline-flex;
justify-content:
center; align-
items: center;
margin: 0 5px;
height: 40px;
width: 40px;
}

</style>
</head>
<body>
<h2>Welcome To Skill/Job Recommender Application</h2>
<div class="container" id="container">
  <div class="form-container sign-up-container">
    <form action="#">
      <h1>Create Account</h1>
      <div class="social-container">
        <a href="#" class="social"><i class="fab fa-
linkedin-in"></i></a>
      </div>
    </form>
  </div>
</div>

```

```

        <span>or use your email for registration</span>
        <input type="text" placeholder="Name" />
        <input type="email" placeholder="Email" />
        <input type="password" placeholder="Password" />
        <button>Sign Up</button>
    </form>
</div>
<div class="form-container sign-in-container">
    <form action="#">
        <h1>Sign in</h1>
        <div class="social-container">
            <a href="#" class="social"><i class="fab fa-
linkedin-in"></i></a>
        </div>
        <span>or use your account</span>
        <input type="email" placeholder="Email" />
        <input type="password"
placeholder="Password" />
        <a href="#">Forgot your password?</a>
        <button><a href="/ApplyJob.html"
style="color:
#fff;">Sign In</button></a>
    </form>
</div>
<div class="overlay-container">
    <div class="overlay">
        <div class="overlay-panel overlay-left">
            <h1>Welcome Back!</h1>
            <p>To keep connected with us please login
with
your personal info</p>
            <button class="ghost" id="signIn">Sign
In</button>
        </div>
        <div class="overlay-panel overlay-right">
            <h1>Hello, Friend!</h1>

```

```

        <p>Enter your personal details and start
journey
with us</p>
        <button class="ghost" id="signUp">Sign
Up</button>
    </div>
</div>
</div>
</div>

<script>
    const signUpButton = document.getElementById('signUp');
    const signInButton = document.getElementById('signIn');
    const container = document.getElementById('container');

    signUpButton.addEventListener('click', () => {
        container.classList.add("right-panel-active");
    });

    signInButton.addEventListener('click', () => {
        container.classList.remove("right-panel-active");
    });
</script>
</body>
</html>

```

7.2 Apply jobs:

```

<!DO
CTYP
E
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>Apply For Job</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4
.
6.2/dist/css/bootstrap.min.css"
integrity="sha384-
xOolHFLEh07PJGoPkLv1IbcEPTNtaed2xpHs
D
9ESMhqIYd0nLMwNLD69Npy4HI+N"
crossorigin="anonymous">
</head>
<body style="padding: 2rem 0;">
    <div class="container">
        <h1 style="text-align: center;">Apply For
Job</h1><br>
        <div class="row mx-0 justify-content-
center">
            <div class="col-md-7 col-lg-5 px-lg-2
col-xl-4 px-xl-0 px-xxl-3">
                <form
method="POST"
class="w-100 rounded-1 p-4 border
bg-
white"
action=""
enctype="multipart/form-data"
>
                    <label class="d-block mb-4">
                        <span class="form-label d-
block">Your name</span>

```



```

        <input                                required
        name="name"
type="text"                                class="form-control"
        placeholder="Enter Your
Name"

        />
    </label>

```

```

        <label class="d-block mb-4">
            <span class="form-label d-
block">Email address</span>
            <input                                required
            name="email"
type="email"                                class="form-control"
            placeholder="Enter your mail
id"

            />
        </label>

```

```

        <label class="d-block mb-4">
            <span class="form-label d-
block">Years of experience</span>
            <select required name="experience"
class="form-select">
                <option>Less than a year</option>
                <option>1 - 2 years</option>
                <option>2 - 4 years</option>
                <option>4 - 7 years</option>
                <option>7 - 10 years</option>
                <option>10+ years</option>
            </select>
        </label>

```

```

        <label class="d-block mb-4">

```

```

        <span class="form-label d-
block">Tell us more about yourself</span>
        <textarea
name="message"
class="form-control"
rows="3"
        placeholder="What motivates you?"
        ></textarea>
    </label>

```

```

    <label class="d-block mb-4">
        <span class="form-label d-
block">Your CV</span>
        <input required name="cv"
type="file" class="form-control" />
    </label>

```

```

    <div class="mb-4">
        <div>
            <div class="form-check">
                <label class="d-block">
                    <input
type="radio"
check-input"
                    class="form-
                    name="remote"
                    value="yes"
                    checked
                    />
                    <span class="form-check-label"
                    >You'd like to work
remotely</span>
                >
            </label>
        </div>
    </div>

```

```

        </div>
        <div>
            <label class="form-check">
                <input
type="radio"                                class="form-
check-input"
name="remote"
value="no"
                />
                <span class="form-check-
label">You'd prefer to work onsite</span>
            </label>
        </div>
    </div>

    <div class="mb-3">
        <button type="submit" class="btn
btn-
primary px-3 rounded-3">
            Apply
        </button>
    </div>

</form>
</div>
</div>
</div>
</body>
</html>

```

7.3 view jobs:

```
<!DOCTYPE
PE 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>View Job</title>
            <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist
/css/bootstrap.min.css" integrity="sha384-
xOolHFLEh07PJGoPkLv1IbcEPTNtaed2xpHsD9ESM
hqIY
d0nLMwNLD69Npy4HI+N" crossorigin="anonymous">

            <style>

                .div {                margin:
                15px;

                }

                #marginbetweennavigationbarandjumbotron {
                margin-top: 10px;                justify-content: center;
                background-color: white;                color: black;
                border-radius: 6px;
                padding: 10px;
                margin-bottom: 20px;
                }

                ul {
```

```
list-style-type: none;
}
</style>
```

```
</head>
```

```
<body>
```

```
<nav class="navbar navbar-default navbar-fixed-top">
  <div class="container-fluid">
    <div class="navbar-header">
      <p><a class="navbar-brand">View Jobs</a></p>
    </div>
```

```
      <!-- End of Navigation and begining of
Jumbotron --
>
```

```
    <div class="container"
id="marginbetweennavigationbarandjumbotron">
      <div class="row">
        <div class="col-xs-12">
```

```
          <h2 class="text-center text-primary"
style="margin:0px;padding:0px;font-
family: 'Droid Sans', sans-serif;">Search Jobs, Across
all India.</h2>
```

```
<hr>
```

```
          <form class="form-group form-inline
textcenter">
```

```

        <input type="text" class="form-
control "
placeholder="Enter Job Title,Designation,Keywords"
style="width:300px;margin:10px;">

```

```

        <select class="form-control"
style="width:300px;margin:10px;">
        <option>Select Experience</option>
        <option>0 Year | Fresher</option>
        <option>1 Year</option>
        <option>2 Years</option>
        <option>3 Years</option>
        <option>4 Years</option>
        <option>5 Year</option>
        <option>6 Years</option>
        <option>7 Years</option>
        <option>8 Years</option>
        <option>9 Years</option>
        <option>10 Years</option>
        <option>More than 10 Years</option>
        </select>
        <p class="text-center">
        <button type="submit" class="btn-
primary text-center form-control"

```

```

style="width:300px;">Search</button>

```

```

        </p>

```

```

    </form>

```

```

    <p class="text-center text-primary">We

```

```

    Will

```

```

    Search Best Matching Jobs for You.</p>

```

```

        </div>
    </div>
</div>
</body>

</html>

```

7.4 Chatbot:

Chatbot is implemented to provide assistance to the users.

```

<script>
    window.watsonAssistantChatOptions = {  integrationID: "f6cbb1f6-
a170-4223-b431-7f1a399ae9ec", // The ID of this integration.  region:
"au-syd", // The region your integration is hosted in.
    serviceInstanceID: "fc1cc8e4-343e-413e-b142-b035f866ec02", //
The ID of your service instance.
    onLoad: function(instance) { instance.render(); }
};
setTimeout(function(){
    const t=document.createElement('script');
t.src="https://webchat.global.assistant.watson.appdomain.cl
oud/versions/"
(window.watsonAssistantChatOptions.clientVersion
'latest') +
"/WatsonAssistantChatEntry.js";
    document.head.appendChild(t);
});
</script>
>

```

7.5 Send grid:

i
m
p
o
r
t
r
e

```
from flask import Flask, render_template, request, redirect,
url_for, session import ibm_db
from flask_mail import Mail, Message
# print(conn)
# print("success")

app = Flask(__name__)

app.secret_key = 'a' conn =
ibm_db.connect("DATABASE=bludb;HOSTNAME=19af6446-
6171-4641-8aba-
9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PO
R
T=30699;SECURITY=SSL;SSLServerCertificate=DigiCertGlo
balRo otCA.crt;UID=cly12896;PWD=Q5dKJaYrZ4llJrzF", "", "")

@app.route('
/') def home():
    return render_template('registration and login.html')

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



```

userid      msg
= "

        if request.method == 'POST':            username =
request.form['username']            password = request.form['password']
sql = "SELECT * FROM users WHERE username=? AND
        password=?"
        stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, username)
ibm_db.bind_param(stmt, 2, password)
ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
print(account)            if account:
session['loggedin'] = True            session['id']
= account['USERNAME']            userid =
account['USERNAME']
session['username'] = account['USERNAME']
        msg = 'Logged in succesfully'

        return render_template('ViewJob.html',
msg=msg)            else:
        msg = 'Incorrect username/password!'            return
render_template('registration and login.html', msg=msg)

```

```

@app.route('/register', methods=['GET',
'POST'])            def register():            msg = "
        if request.method == 'POST':
username = request.form['username']
email = request.form['email']            password =
request.form['password']
        sql = "SELECT * FROM users WHERE username=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
print(account)            if account:            msg =

```

```

'Account Already exists'      elif not
re.match(r'^[@ ]+@[^@]+\.[^@]+', email):
    msg = 'Invalid email'      elif not re.match(r'[A-
Za-z0-9]+', username):      msg = 'name must contain only
alpha characters or numbers!' else:      insert_sql =
"INSERT INTO users VALUES(?,?,?)"      prep_stmt =
ibm_db.prepare(conn, insert_sql)
ibm_db.bind_param(prepare_stmt, 1, username)
ibm_db.bind_param(prepare_stmt, 2, email)
ibm_db.bind_param(prepare_stmt, 3, password)
ibm_db.execute(prepare_stmt)      msg = 'you have
successfully registered'      app.config['SECRET_KEY'] =
'top-secret!'      app.config['MAIL_SERVER'] =
'smtp.sendgrid.net'      app.config['MAIL_PORT'] = 587
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USERNAME'] = 'apikey'
app.config['MAIL_PASSWORD'] =
'SG.XbHqaobAQPCL5ZW_3-jRkA.vuaAYWQBDNuRV-
5MjIVERohpOKt-dKvcQmXNsgjFi74'

app.config['MAIL_DEFAULT_SENDER'] =
'jasperkirubakaranjit2019@citchennai.net'
mail = Mail(app)
    recipient = request.form['email']
    msg = Message('Successfully Registered',
recipients=[recipient])      msg.body =
('Congratulations! You have successfully registered
with '
        'Skill/Job Recommender!')
    msg.html = ('<h1>Successfully Registered</h1>'
        '<p>Congratulations! You have successfully
registered with '
        '<b>Skill/Job Recommender</b>!</p>')
    mail.send(msg)      return
render_template('registration and login.html')      elif
request.method == 'POST':      msg = 'Please fill out the

```

```
form!'          return render_template('registration and  
login.html', msg=msg)
```

```
    @app.route('/vie  
w')          def view():  
        return render_template('ViewJob.html')
```

```
    @app.route('/apply', methods=['GET',  
'POST'])    def apply():          msg = "  
        if request.method == 'POST':  
username = request.form['username']  
email = request.form['email']  
        qualification                    =  
request.form['qualification']            skills =  
request.form['skills']                   jobs  =  
request.form['jobs']
```

```
        insert_sql = "INSERT INTO jobs  
values(?,?,?,?,?)"      prep_stmt =  
ibm_db.prepare(conn, insert_sql)  
ibm_db.bind_param(prepare_stmt, 1, username)  
ibm_db.bind_param(prepare_stmt, 2, email)  
ibm_db.bind_param(prepare_stmt, 3, qualification)  
ibm_db.bind_param(prepare_stmt, 4, skills)  
ibm_db.bind_param(prepare_stmt, 5, jobs)  
ibm_db.execute(prepare_stmt)      msg = "You have  
succesfully applied for the job!"  
        session['loggedin'] = True  
        app.config['SECRET_KEY'] = 'top-secret!'  
        app.config['MAIL_SERVER'] = 'smtp.sendgrid.net'  
app.config['MAIL_PORT'] = 587  
app.config['MAIL_USE_TLS'] = True  
app.config['MAIL_USERNAME'] = 'apikey'  
app.config['MAIL_PASSWORD'] =
```

```

'SG.XbHqaobAQPCL5ZW_3-jRkA.vuaAYWQBDNuRV-
5MjIVERohpOKt-dKvcQmXNsgjFi74'
    app.config['MAIL_DEFAULT_SENDER'] =
'jasperkirubakaranjit2019@citchennai.net'    mail =
Mail(app)    recipient = request.form['email']    msg
= Message('Successfully Applied',
recipients=[recipient])    msg.body = ('Congratulations! You
have successfully applied
    your job with '
        'Skill/Job Recommender!')
    msg.html = ('<h1>Successfully Applied</h1>'
        '<p>Congratulations! You have successfully
applied your job with '
        '<b>Skill/Job Recommender</b>!</p>')
mail.send(msg)
    return redirect(url_for('login'))

    elif request.method == 'POST':    msg =
        'Please fill the form!'
    return render_template('ApplyJob.html', msg=msg)

@app.route('/logou
t') def logout():
    session.pop('loggedin', None)
session.pop('id', None)    session.pop('username',
None)    return render_template('registration and
login.html')

```

8. Testing:

8.1 Test cases:

Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
Login page	Verify that after registration users are navigated to login page	Mail id, Username, Password, Phone number, Pin	1. Open the website and go to register page. 2. Enter details and press register 3. Verify that users are navigated to registration page	https://drive.google.com/file/d/14ri8o8b0iK7TARZE1x7ZK4WvwMRK083o/view?usp=share_link	Users should be navigated to registration page	Working as expected	Pass	Excellent	N		Ashwin Kumar P
Home Page	Verify the UI elements in Login/Signup popup	Username & Password	1. Open the website 2. Enter details and press login 3. Verify that users are notified of login process	https://drive.google.com/file/d/1uAPmwLs6im6ktaeu0uMa7oCqWXPfK/view?usp=share_link	Users should be notified of login process	Working as expected	Pass	Good	N	BUG-12	Karthik P
Home page	Verify user is able to log into application with Valid credentials		1. Open the website 2. Enter details and press login 3. Verify that users are logged into website properly	Username: karthickpremoth@gmail.com password: 123	User should be logged into website properly	Working as expected	Pass	Good	N		Nandakumar R
Home Page	Verify that categories of skills and jobs are shown in homepage		1. Open the website 2. Enter details and press login 3. Verify that categories of are showing Jobs shown in	https://drive.google.com/file/d/1mabbKQ2zQ3X0zSu6BCEaWwmfHmDgboF/view?usp=share_link	Categories of skills and jobs should be shown in homepage	Working as expected	Pass	Good	N	BUG-14	Karthik P
Home page	Verify that jobs are displayed in homepage		1. Open the website 2. Enter details and press login 3. Verify that jobs are displayed in homepage	https://drive.google.com/file/d/16G13GY925DK2xeie0sE6Bfk-LkLDrtv/view?usp=share_li	Jobs should be displayed in homepage	Working as expected	Pass	Good	N		Jasper Kirubakaran J
Home page	Verify that when clicked on jobs it is redirected to correct page		1. Open the website 2. Enter details and press login 3. Verify that when clicked on jobs it is redirected to apply job	https://drive.google.com/file/d/11OvuOhHUE3abENAJ6OC8Jg14ti-icacE/view?usp=sharing	When clicked on apply job link it should be submitted and email is sent to the user, after that user can able to logout.	Working as expected	Pass	Excellent	N		Karthik P

8.2 User acceptance testing:

1. Purpose of Document

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

2. 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	3	2	1	1	7
Duplicate	1	0	2	0	3
External	2	0	0	1	3
Fixed	5	2	5	7	19
Not Reproduced	0	0	1	0	1
Skipped	0	0	0	1	1
Won't Fix	0	5	1	1	7
Totals	11	9	10	11	41

3. Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	5	0	0	5
Security	3	0	0	3
Outsource Shipping	7	0	0	7

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

9.Results:

9.1 Performance metrics:

The project has been completed and executed as we expected. All the details were stored in the IBM DB2 and the app was deployed to the cloud. We ensured that the database was well connected using python flask and details were stored. the motive of recommending jobs to the seekers were implemented and was testing by means of various testing methodologies and so expected outcome was obtained.

10. Advantages and disadvantages:

Advantages:

1. Based on their skill set, a person looking for work can quickly locate a position that suits them.
2. A person can take an eligibility exam to determine their eligibility.
3. The majority of recruiters choose the right candidate based on the scores they received on the eligibility tests.

Disadvantages:

1. Job of Person While taking the eligibility test, there can be technical issues.
2. Job seekers may find it difficult to get in touch with recruiters directly.
3. Irrelevant jobs will be suggested if data is not sufficient.
4. The web application will take more time to load if it is facing huge requests.

11. Conclusion:

The application was created to simplify the job hunt. We created an application that is easy to use. based on their skill set, a user can quickly obtain employment. The use of this application benefits the jobseeker without a doubt. Additionally, chatbot is implemented and integrated which provides guidance for the job seekers to apply for the job.

12.Future scope:

The popular programme linked in allows users to look for work and maintain connections with colleagues and organisations

Employers and job seekers both use LinkedIn to locate employment. There are many opportunities to improve our project in the future in a manner similar to linked in.

13.Appendix:

13.1 Source code:

[Sprint1: Registration & login.html:](#)

```
<!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,
initialscale=1.0">

  <title>Registration and Login</title>
  <link rel="stylesheet" href="./style.css"/>
  <script defer
src="https://use.fontawesome.com/releases/v5.15.4/js/all.js"
integrity="sha384-
rOA1PnstxnOBLzCLMcre8ybwbTmemjzdNiILg8O7z1lUkLXozs4D
Hon lDtnE7fpc" crossorigin="anonymous"></script>
  <style>

@import
url('https://fonts.googleapis.com/css?family=Montserrat:400,800');
```

```
* {  
    box-sizing: border-box;  
}
```

```
body {  
    background: #f6f5f7;  
    display: flex;    justify-  
content: center; align-items: center;  
    flex-direction: column;  
    font-family: 'Montserrat',  
sans-serif;  
    height: 100vh;  
    margin: -20px 0 50px;  
}
```

```
h1 { font-weight:  
    bold;  
    margin: 0;  
}
```

```
h2 {  
    text-align: center;  
}
```

```
p {  
    font-size: 14px;  
    font-weight: 100;  
    line-height: 20px;  
    letter-spacing: 0.5px;  
    margin: 20px 0 30px;  
}
```

```
span {  
    font-size:  
    12px;  
}
```

```
a {  
    color: #333;  
    font-size: 14px;  
    text-decoration:  
    none;  
    margin: 15px 0;  
}
```

```
button {    border-radius:  
20px;    border: 1px  
solid #41d3ff;  
    background-color:  
#41d3ff; color: #FFFFFF;  
    font-size: 12px;  
    font-weight: bold;  
    padding: 12px 45px;  
    letter-spacing: 1px;  
    text-transform:  
uppercase;  
    transition: transform 80ms ease-in;  
}
```

```
button:active {  
    transform: scale(0.95);  
}
```

```
button:focus {  
    outline: none;
```

```
}
```

```
button.ghost { background-  
color: transparent;  
    border-color: #FFFFFF;  
}
```

```
form { background-  
color: #FFFFFF;  
    display: flex; align-  
items: center; justify-  
content: center; flex-  
direction: column;  
    padding: 0 50px;  
    height: 100%;  
    text-align: center;  
}
```

```
input {  
    background-color:  
#eee;    border: none;  
    padding: 12px  
15px;    margin: 8px  
0;  
    width: 100%;  
}
```

```
.container { background-color: #fff;  
    border-radius: 10px;    box-  
shadow: 0 14px 28px rgba(0,0,0,0.25),  
        0 10px 10px  
rgba(0,0,0,0.22);  
    position: relative;  
    overflow: hidden;
```

```
    width: 768px;
    max-width: 100%;
    min-height: 480px;
}
```

```
.form-container {
    position:
absolute; top: 0;
    height: 100%;
    transition: all 0.6s ease-in-out;
}
```

```
.sign-in-
container {
    left: 0;
    width:
50%;
    z-index: 2;
}
```

```
.container.right-panel-active .sign-in-container {
    transform: translateX(100%);
}
```

```
.sign-up-
container {
    left: 0;
    width:
50%;
    opacity:
0; z-index:
1;
```

```
}
```

```
.container.right-panel-active .sign-up-container {  
    transform: translateX(100%);  
    opacity: 1;  
    z-index: 5;  
    animation: show 0.6s;  
}
```

```
@keyframes show  
{  
    0%, 49.99%  
{  
        opacity:  
0;        z-index:  
1;  
    }  
  
    50%, 100% {  
        opacity: 1;  
        z-index: 5;  
    }  
}
```

```
.overlay-container  
{  
    position:  
absolute; top: 0;  
    left: 50%;  
    width: 50%;  
    height:  
100%;  
    overflow:  
hidden;  
    transition: transform 0.6s ease-in-out;  
    z-index: 100;  
}
```

```
.container.right-panel-active .overlay-container{
    transform: translateX(-100%);
}
```

```
.overlay {
    background: #41d3ff;
    background: -webkit-linear-gradient(to right, #41d3ff,
#FF416C);    background: linear-gradient(to right, #41d3ff,
#FF416C);
    background-repeat: no-
repeat;    background-size:
cover;    background-
position: 0 0;    color:
#FFFFFF;    position:
relative;    left: -100%;
    height: 100%;
    width: 200%;
    transform: translateX(0);
    transition: transform 0.6s ease-in-out;
}
```

```
.container.right-panel-active .overlay {
    transform: translateX(50%);
}
```

```
.overlay-panel {
    position:
absolute; display:
flex;    align-items:
center;    justify-
content: center; flex-
direction: column;
```

```
padding: 0 40px;
text-align: center;
top: 0;
height: 100%;
width: 50%;
transform: translateX(0);
transition: transform 0.6s ease-
in-out;
}
```

```
.overlay-left {
  transform: translateX(-20%);
}
```

```
.container.right-panel-active .overlay-left {
  transform: translateX(0);
}
```

```
.overlay-right {
  right: 0; transform:
translateX(0);
}
```

```
.container.right-panel-active .overlay-right {      transform:
translateX(20%);
}
```

```
.social-container {
  margin: 20px 0;
}
```



```

.social-container a {
    border: 1px solid
#DDDDDD;    border-
radius: 50%;    display:
inline-flex;    justify-
content: center; align-items:
center;    margin: 0 5px;
    height: 40px;
    width: 40px;
}

</style>
</head>
<body>
    <h2>Welcome To Skill/Job Recommender Application</h2>
    <div class="container" id="container">
        <div class="form-container sign-up-container">
            <form action="#">
                <h1>Create Account</h1>
                <div class="social-container">
                    <a href="#" class="social"><i class="fab fa-
linkedinin"></i></a>
                </div>
                <span>or use your email for registration</span>
                <input type="text" placeholder="Name" />
                <input type="email" placeholder="Email" />
                <input type="password" placeholder="Password" />
                <button>Sign Up</button>
            </form>
        </div>
        <div class="form-container sign-in-container">
            <form action="#">
                <h1>Sign in</h1>
                <div class="social-container">

```

```

        <a href="#" class="social"><i class="fab fa-
linkedinin"></i></a>
    </div>
    <span>or use your account</span>
    <input type="email" placeholder="Email" />
    <input type="password" placeholder="Password" />
    <a href="#">Forgot your password?</a>
    <button><a href="./ApplyJob.html" style="color:
#fff;">Sign
In</button></a>
    </form>
</div>
<div class="overlay-container">
    <div class="overlay">
        <div class="overlay-panel overlay-left">
            <h1>Welcome Back!</h1>
            <p>To keep connected with us please login with your
personal info</p>
            <button class="ghost" id="signIn">Sign In</button>
        </div>
        <div class="overlay-panel overlay-right">
            <h1>Hello, Friend!</h1>
            <p>Enter your personal details and start journey with
us</p>
            <button class="ghost" id="signUp">Sign Up</button>
        </div>
    </div>
</div>
</div>
</div>
</div>
</div>
<script>
    const signUpButton =
document.getElementById('signUp'); const signInButton
= document.getElementById('signIn'); const container =
document.getElementById('container');

```

```

signUpButton.addEventListener('click', () => {
    container.classList.add("right-panel-active");
});

signInButton.addEventListener('click', () => {
    container.classList.remove("right-panel-active");
});
</script>
</body>
</html>

```

Apply jobs.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,
initialscale=1.0">
  <title>Apply For Job</title>
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/css/bootstra
p.min.css" integrity="sha384-
xOolHFLEh07PJGoPkLv1IbcEPTNtaed2xpHsD9ESMhqIYd0nLM
wN
LD69Npy4HI+N" crossorigin="anonymous">
</head>
<body style="padding: 2rem 0;">
  <div class="container">
    <h1 style="text-align: center;">Apply For Job</h1><br>
    <div class="row mx-0 justify-content-center">
      <div class="col-md-7 col-lg-5 px-lg-2 col-xl-4 px-xl-0 px-
xxl3">

```

```

    <form
method="POST"
    class="w-100 rounded-1 p-4 border bg-white"
action=""
    enctype="multipart/form-data"
>
    <label class="d-block mb-4">
        <span class="form-label d-block">Your name</span>
        <input
required
name="name"
type="text"
class="form-control"
        placeholder="Enter Your Name"
        />
    </label>

    <label class="d-block mb-4">
        <span class="form-label d-block">Email address</span>
        <input
required
name="email"
type="email"
class="form-control"
        placeholder="Enter your mail id"
        />
    </label>

    <label class="d-block mb-4">
        <span class="form-label d-block">Years of
experience</span>
        <select required name="experience" class="form-select">
            <option>Less than a year</option>
            <option>1 - 2 years</option>
            <option>2 - 4 years</option>
            <option>4 - 7 years</option>
            <option>7 - 10 years</option>

```

```

        <option>10+ years</option>
    </select>
</label>

<label class="d-block mb-4">
    <span class="form-label d-block">Tell us more about
yourself</span>
    <textarea
name="message"
class="form-control"
    rows="3"
    placeholder="What motivates you?"
    ></textarea>
</label>

<label class="d-block mb-4">
    <span class="form-label d-block">Your CV</span>
    <input required name="cv" type="file" class="form-
control" />
</label>

<div class="mb-4">
    <div>
        <div class="form-check">
            <label class="d-block">
                <input
type="radio"
                class="form-check-input"

name="remote"
value="yes"
checked

                />
                <span class="form-check-label"
                >You'd like to work remotely</span>
            >

```

```

        </label>
    </div>
</div>
<div>
    <label class="form-check">
        <input
type="radio"
        class="form-check-
input"
name="remote"
value="no"
        <span class="form-check-label">You'd prefer to work
onsite</span>
    </label>
</div>
</div>

<div class="mb-3">
    <button type="submit" class="btn btn-primary px-3
rounded3">
        Apply
    </button>
</div>

</form>
</div>
</div>
</div>
</body>
<html lang="en">

<head>
    <meta charset="UTF-8">

```

```
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width,
initialscale=1.0">
<title>View Job</title>
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/css/bootstra
p.min.css" integrity="sha384-
xOolHFLEh07PJGoPkLv1IbcEPTNtaed2xpHsD9ESMhqIYd0nLM
wN
LD69Npy4HI+N" crossorigin="anonymous">
```

```
<style>
.div {
    margin: 15px;
}

#marginbetweennavigationbarandjumbotron {
    margin-top: 10px;
    justify-content: center;
    background-color: white;
    color: black;
    border-radius: 6px;
    padding: 10px;
    margin-bottom: 20px;
}
ul
{
    list-style-type: none;
}
</style>
```

```
</head>
```

[View jobs.html:](#)

```

<!doctype html>
<body>

    <nav class="navbar navbar-default navbar-fixed-top">
        <div class="container-fluid">
            <div class="navbar-header">
                <p><a class="navbar-brand">View Jobs</a></p>
            </div>

        <!-- End of Navigation and begining of Jumbotron -->

        <div class="container"
id="marginbetweennavigationbarandjumbotron">
            <div class="row">
                <div class="col-xs-12">

                    <h2 class="text-center text-primary"
style="margin:0px;padding:0px;font-family: 'Droid
Sans', sans-serif;">Search Jobs, Across
all India.</h2>
                    <hr>
                    <form class="form-group form-inline text-center">
<input type="text" class="form-control " placeholder="Enter Job
Title,Desigation,Keywords"
style="width:300px;margin:10px;">
                    <select class="form-control"
style="width:300px;margin:10px;">
                        <option>Select Experience</option>
                        <option>0 Year | Fresher</option>
                        <option>1 Year</option>
                        <option>2 Years</option>
                        <option>3 Years</option>

```



```

        <option>4 Years</option>
        <option>5 Year</option>
        <option>6 Years</option>
        <option>7 Years</option>
        <option>8 Years</option>
        <option>9 Years</option>
        <option>10 Years</option>
        <option>More than 10 Years</option>
    </select>
    <p class="text-center">
        <button type="submit" class="btn-primary
textcenter form-control"
            style="width:300px;">Search</button>
    </p>
</form>
    <p class="text-center text-primary">We Will Search
Best Matching Jobs for You.</p>

```

```

        </div>
    </div>
</div>
</body>

```

Sprint 2 Integrating chatbot:

Javascript:

```

<script>
    window.watsonAssistantChatOptions = {      integrationID:
"f6cbb1f6-a170-4223-b431-7f1a399ae9ec", // The ID of this
integration.
        region: "au-syd", // The region your integration is hosted in.
        serviceInstanceID: "fc1cc8e4-343e-413e-b142-b035f866ec02", //
The ID of your service instance.
    };

```

```

    onLoad: function(instance) { instance.render(); }
  };
  setTimeout(function(){
    const t=document.createElement('script');
    t.src="https://webchat.global.assistant.watson.appdomain.cloud/versions/"
    (window.watsonAssistantChatOptions.clientVersion
    'latest') +
    "/WatsonAssistantChatEntry.js";
    document.head.appendChild(t);
  });
</script>

```

Sprint 3 Integrating send grid:

```

import re
from flask import Flask, render_template, request, redirect,
url_for, session import ibm_db
from flask_mail import Mail, Message
# print(conn)
# print("success")

app = Flask(__name__)

app.secret_key = 'a'
conn =
ibm_db.connect("DATABASE=bludb;HOSTNAME=19af6446-
6171-4641-8aba-
9dcff8e1b6ff.clogj3sd0tgu0lqde00.databases.appdomain.cloud;PO
RT
=30699;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRo
otC
A.crt;UID=cly12896;PWD=Q5dKJaYrZ4lJrzF", "", "")

@app.route('/') def home():    return render_template('registration
and login.html')

```

```

@app.route('/login', methods=['GET',
'POST']) def login():    global userid
    msg = "

    if request.method == 'POST':        username =
request.form['username']
        password = request.form['password']
        sql = "SELECT * FROM users WHERE username=? AND
password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)        account =
        ibm_db.fetch_assoc(stmt)
        print(account)        if account:
session['loggedin'] = True        session['id'] =
account['USERNAME']        userid =
account['USERNAME']
session['username'] = account['USERNAME']
        msg = 'Logged in succesfully'

        return render_template('ViewJob.html',
msg=msg)        else:
            msg = 'Incorrect username/password!'
            return render_template('registration and login.html', msg=msg)
@app.route('/register', methods=['GET', 'POST']) def register():
msg = "    if request.method == 'POST':        username =
request.form['username']        email = request.form['email']
        password = request.form['password']
        sql = "SELECT * FROM users WHERE username=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1,
username)        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)

        print(account)
        if account:

```

```

        msg = 'Account Already exists' elif not
re.match(r'^@]+@^[^@]+\.[^@]+', email):
        msg = 'Invalid email' elif not re.match(r'[A-Za-z0-
9]+', username): msg = 'name must contain only alpha
characters or numbers!' else:
        insert_sql = "INSERT INTO users
VALUES(?,?,?)" prep_stmt =
ibm_db.prepare(conn, insert_sql)
ibm_db.bind_param(prepare_stmt, 1, username)
ibm_db.bind_param(prepare_stmt, 2, email)
ibm_db.bind_param(prepare_stmt, 3, password)
ibm_db.execute(prepare_stmt) msg = 'you have
successfully registered'
app.config['SECRET_KEY'] = 'top-secret!'
app.config['MAIL_SERVER'] = 'smtp.sendgrid.net'
app.config['MAIL_PORT'] = 587
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USERNAME'] = 'apikey'
app.config['MAIL_PASSWORD'] =
'SG.XbHqaobAQPCL5ZW_3-jRkA.vuaAYWQBBDNuRV-
5MjIVERohpOKt-dKvcQmXNsgjFi74'
app.config['MAIL_DEFAULT_SENDER'] =
'jasperkirubakaranjit2019@citchennai.net
'
        mail = Mail(app)
        recipient = request.form['email']
msg = Message('Successfully Registered',
recipients=[recipient])
        msg.body = ('Congratulations! You have successfully
registered with '
                    'Skill/Job Recommender!')
        msg.html = ('<h1>Successfully Registered</h1>'
                    '<p>Congratulations! You have successfully
registered with '
                    '<b>Skill/Job
Recommender</b>!</p>') mail.send(msg)
return render_template('registration and login.html')

```

```

elif request.method == 'POST': msg = 'Please fill out
the form!'
    return render_template('registration and login.html', msg=msg)
@app.route('/view') def view():    return
render_template('ViewJob.html')

```

```

@app.route('/apply', methods=['GET',
'POST']) def apply():    msg = "    if
request.method == 'POST':
username = request.form['username']
email = request.form['email']
        qualification                =
request.form['qualification']        skills =
request.form['skills']                jobs =
request.form['jobs']

```

```

        insert_sql = "INSERT INTO jobs values(?,?,?,?,?)"
prep_stmt = ibm_db.prepare(conn, insert_sql)
ibm_db.bind_param(prepare_stmt, 1, username)
ibm_db.bind_param(prepare_stmt, 2, email)
ibm_db.bind_param(prepare_stmt, 3, qualification)
ibm_db.bind_param(prepare_stmt, 4, skills)
ibm_db.bind_param(prepare_stmt, 5, jobs)
ibm_db.execute(prepare_stmt)
        msg = "You have succesfully applied for the job!"
        session['loggedin'] = True
        app.config['SECRET_KEY'] = 'top-secret!'
app.config['MAIL_SERVER'] = 'smtp.sendgrid.net'
app.config['MAIL_PORT'] = 587
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USERNAME'] = 'apikey'
        app.config['MAIL_PASSWORD'] =
'SG.XbHqaobAQPCL5ZW_3-jRkA.vuaAYWQBBDNuRV-

```

```

5MjIVERohpOKt-dKvcQmXNsgjFi74'
app.config['MAIL_DEFAULT_SENDER'] =
'jasperkirubakaranjit2019@citchennai.net'
    mail = Mail(app)
    recipient = request.form['email']
msg = Message('Successfully
Applied',
recipients=[recipient])
    msg.body = ('Congratulations! You have successfully applied
your job with '
                'Skill/Job Recommender!')
    msg.html = ('<h1>Successfully Applied</h1>'
                '<p>Congratulations! You have successfully applied
your job with '
                '<b>Skill/Job Recommender</b>!</p>')
mail.send(msg)
    return redirect(url_for('login'))

elif request.method ==
'POST':    msg = 'Please fill
the form!'
    return render_template('ApplyJob.html', msg=msg)

```

```

@app.route('/logout') def logout():
session.pop('loggedin', None)    session.pop('id',
None)    session.pop('username', None)    return
render_template('registration and login.html')

```

13.2 Github & project demo link:

All the tasks of implementing the web application is uploaded in our github

Github: [**https://github.com/IBM-EPBL/IBM-Project-50420-1660907803**](https://github.com/IBM-EPBL/IBM-Project-50420-1660907803)

Demo:

[https://drive.google.com/file/d/1se07rhzSjTQCPoM3T34jL_1wBLv5G821 /view?usp=sharing](https://drive.google.com/file/d/1se07rhzSjTQCPoM3T34jL_1wBLv5G821/view?usp=sharing)