

SKILL/JOB RECOMMENDER APPLICATION

Team Leader : Sainath M

Team member : Adith M S

Team member : Adhithya T P

Team member : Karthick V

Team member:Lokesh kumar S

Team ID: PNT2022TMID22478

INTRODUCTION

In this project we are developing a skill job recommender application , used for applying jobs that are available. This job recommender application can be helpful to apply jobs by online method and can be used to get quick alerts about it

LITERATURE SURVEY

MACHINE LEARNED JOB RECOMMENDATION

:<https://dl.acm.org/doi/10.1145/2043932.2043994>

COMBINING CONTENT BASED AND COLLABORATIVE FILTERING FOR JOB RECOMMENDATION:

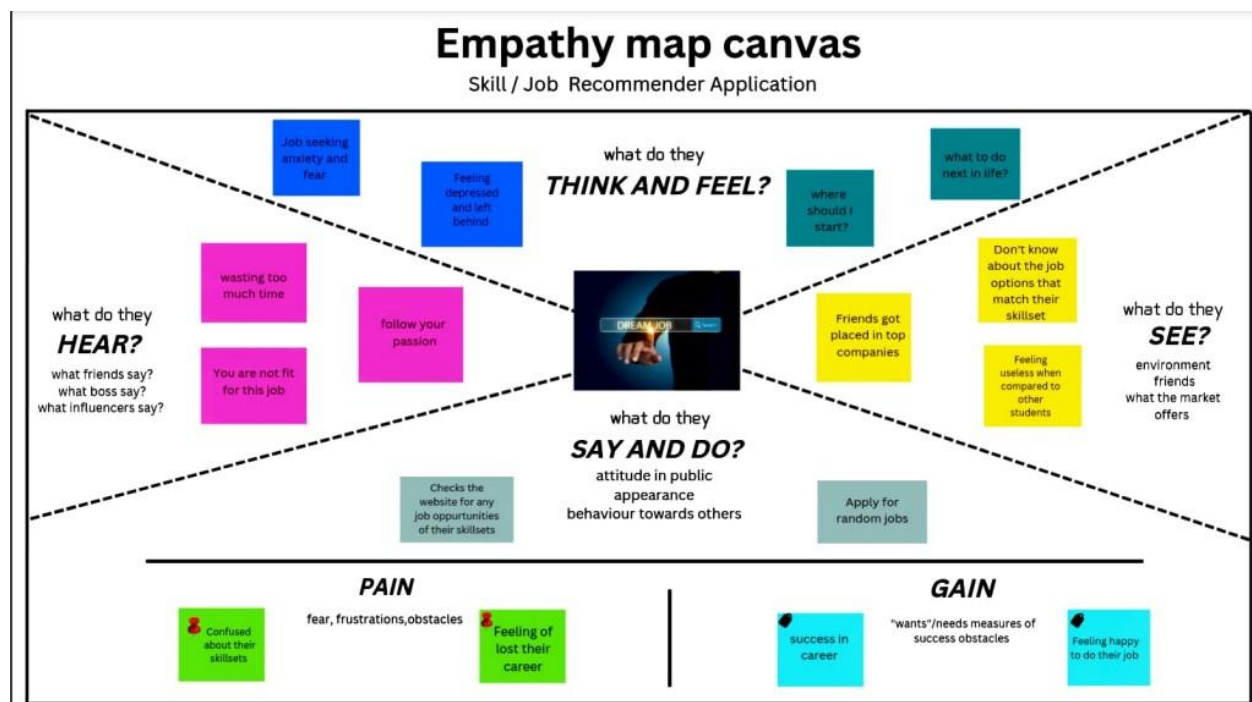
<https://www.sciencedirect.com/science/article/abs/pii/S095070511730374X?via%3Dihub>

PERSONALIZED JOB RECOMMENDATION SYSTEM AT LINKEDIN: <https://dl.acm.org/doi/10.1145/3109859.3109921>

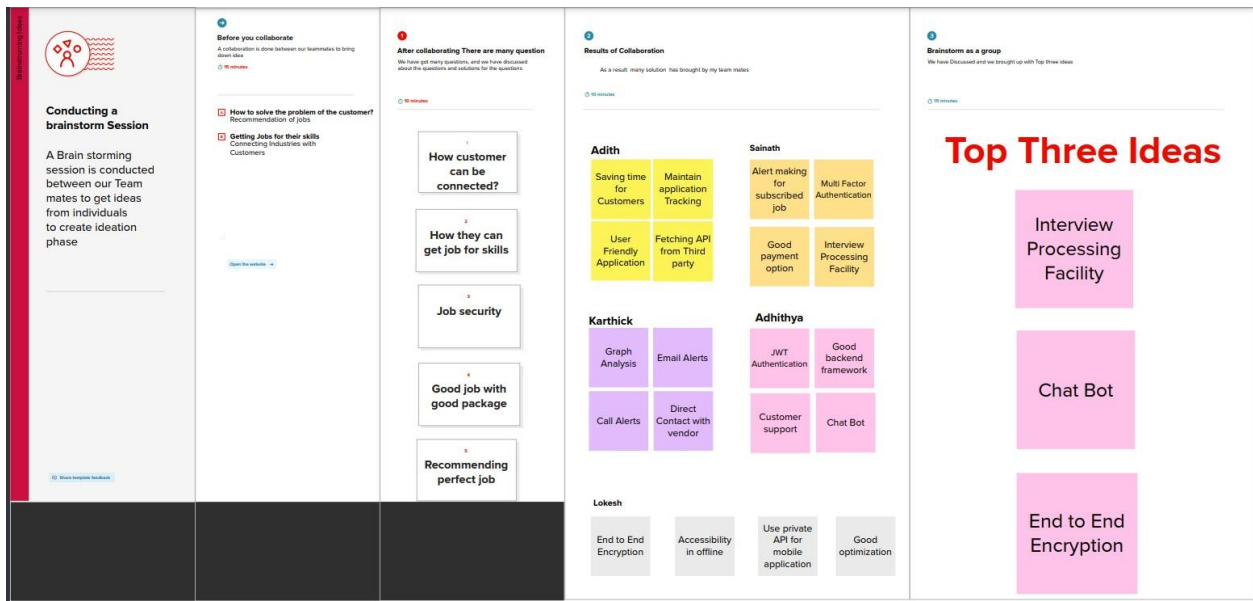
A NOTE ON EXPLICIT VERSUS IMPLICIT INFORMATION FOR JOB RECOMMENDATION:

<https://www.sciencedirect.com/science/article/abs/pii/S0167923617300611?via%3Dihub>

EMPATHY MAP



BRAINSTORM



PROBLEM AND SOLUTION

HOW CUSTOMERS MEET JOB?

The software uses two types of account, one is vendor type another is customer type, so the job posted by vendors can be easily accessed by customers

HOW CUSTOMERS GET SUGGESTIONS?

As the profile created for customers, all the experience and skill sets are gathered, so a special type of

algorithm will provide suggestion about job that will match their profile

HOW CUSTOMERS CLARIFY THEIR PROBLEMS?

The software uses customer support facility and chatter bot, so any questions are clarified both vendor and customer side

WHY JOB RECOMMENDATION APPLICATION?

Many individuals in society are without job due to many reasons so, we come up with online application it is easy to use and all individuals can apply for the job that fit for their skill

TIME AND MONEY?

As it is a online platform, the time and money can be saved, comparing to offline platform

PROPOSED SOLUTION

PROBLEM IN JOB RECOMMENDATION

In Society many individuals face problems in job searching , many job seekers are unable to find their dream

job. Many good technical persons are unable to land their dream job, and lose their hope. So we have come up with a solution.

SOLUTION FOR THE PROBLEM

Our teammates, have designed a multi speciality software which helps job seekers to land in their dream job according to their skills

NOVELTY

This software has designed to get Recommended job for the job seekers, the unique thing in this software is it has two types of account, one is vendor type account, another one is customer type, so The job posted by the vendors can easily meet the customers

FEASIBILITY

The project is feasible and can be implemented using flask framework, and the job api can be brought from third party service, and the software can be accessed from all over the world to meet job at all ends

BUSINESS MODEL

Apart from job recommendation, a revenue is important for a organization, so the required revenue can be brought up by third party ads like google ads

SOCIAL IMPACT

This software solves the social impacts like making all job seekers or individuals to meet the job that meets their criteria, so this can solve social issue on job finding

SCALABILITY

This software is based on SDLC, so the scalability of the software can be changed according to the needs of customers in future

SOLUTION ARCHITECTURE

INTRODUCTION

Every solution architecture design contains 6 to 7 phases these standards should be followed by all development team to ensure the standard of the software, so the software is scalable, versatile and reusable

REQUIREMENTS

This project is done using the **Django framework** for backend development, and other required packages like **allauth, django-form, djangosignals security packages** etc..

For frontend development **css, HTML, javascript** is used along with css framework like **bootstrap**.

For API testing **postman** application is used, and for deployment **IBM cloud service** is used.

DESIGN

All the requirements are used to design the software. The design and architecture of the software is done in a unique manner so the software can be reused and developed in future.

The routers are programmed in **view.py** file, The forms used in the software are developed in **forms.py** file. The database model is created in **model.py** file, the testings are done in separate **tests.py** file. Finally HTML files are stored in **templates folder** and static file is stored in **static folder**

IMPLEMENTATION

The designing process is done and implementation is done by developing the logic by coding. All the required packages are imported and for each router specific logic is developed according to the use.

UNIT TESTING

Each part of the software is developed by individual team members, and it is tested individually by the python unit testing package.

INTEGRATION AND TESTING

After unit testing all parts of the software are integrated and tested finally, so the flask application can be runned in any platform. The testing process includes **Alpha testing** and **Beta testing**.

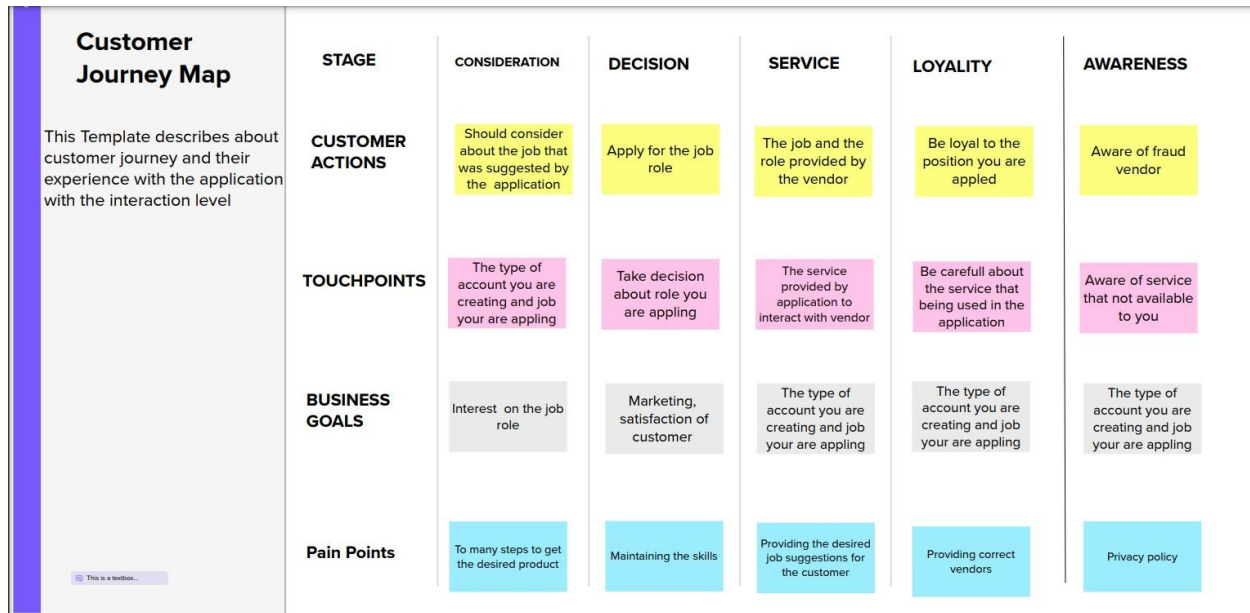
DEPLOYMENT

The flask application is finally deployed in IAAS platform like **IBM cloud service**, so it can be runned in **HTTPS protocol** along with **SSL**. In the deployment process a real time database is connected along with real time file storage.

MAINTENANCE

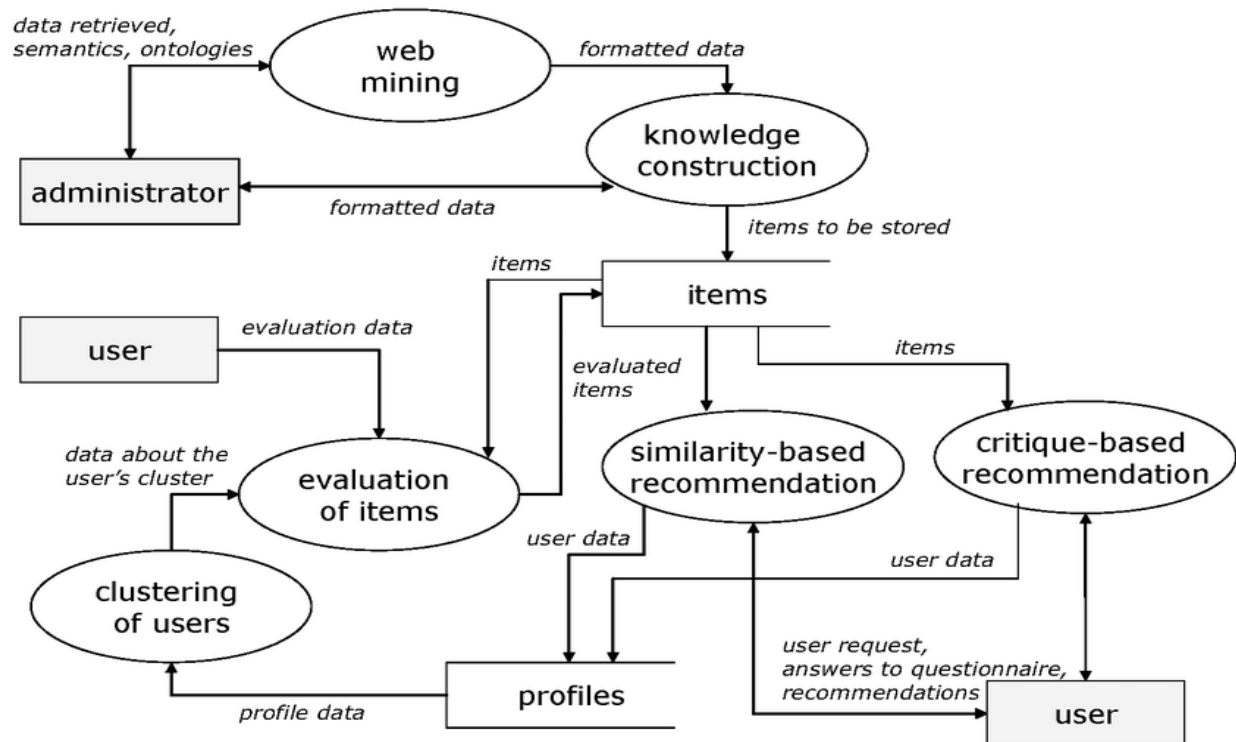
After successful deployment, if there is a package update, it is implemented in the software.

CUSTOMER JOURNEY MAP



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.



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 (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and	I can access my account / dashboard	High	Sprint-1

			confirming my password.			
Customer (Mobile user)	User registration confirmation	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
Customer (Mobile user)	User registration through third party web application	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
Customer (Mobile user)	User registration through third party	USN -4	As a user, I can register for the application through Gmail	with google application	Medium	Sprint -1
Customer (Mobile user)	Login	USN -5	As a user, I can log into the application by entering email & password	login system using flask	High	Sprint -1
Customer (Mobile user)	Dashboard	USN -6	The activities and records of the user are rendered in dashboard	User can read, delete his/her record	High	Sprint -1
Customer (Web user)	Job suggestion	USN -7	The jobs are suggested as per the	from profile page	Low	Sprint -1

			skills mentioned by the user in profile			
Customer Care Executive	Support for customers	USN -8	Doubts and suggestions can be clarified using chatbot and customer care number	Can used by user	High	Sprint -2
Administrator	Superuser	USN -9	All functionalities are managed by superuser called as Administrator	can accessed by specific user	High	Sprint -2

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	1.Registration through Form 2.Registration through Gmail 3.Registration through LinkedIn
FR-2	User Confirmation	1.Confirmation via Email 2.Confirmation via OTP
FR-3	User Login	1.Login through Email and password

		2.Google authenticator used if multifactor authentication is enabled
FR-4	Creating Profile	1.Required data of the user is collected to create the profile for applying job 2.Skills of the user is collected for job suggestion
FR-5	Job suggestion and apply	1.The available jobs are recommended to the user as per the user skills 2.Applying for the job
FR-6	Alerts and Notification	1.Job Alerts are sent to the user email 2.Information and status about the applied job is sent to email

Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	1. Good Interaction with the client side serve 2. Good user experience

		<p>3. Immediate changes can be done in profile as per the job requirement</p>
NFR-2	Security	<p>1. The user account is secured by flask framework and middlewares</p> <p>2.Multifactor authentication is also used for security</p> <p>3. User data is protected against external attack</p>
NFR-3	Reliability	<p>1.More versatile</p> <p>2.Easy to work with the application</p> <p>3.Good UI/UX</p>
NFR-4	Performance	<p>1.The application will have a good performance due to python's flask framework</p> <p>2.Easy routers and their functions are isolated</p> <p>3.Due to SPA, the user can experience good client side experience</p>
NFR-5	Availability	<p>1.The application can available at any region at any time</p> <p>2. It is independent on time and location or IP</p>

NFR-6	Scalability	<p>1.The application is much scalable due to it's framework and development</p> <p>2.The Database used in this application can be scalable according to the use</p> <p>3.And rabid development can be done according to the customer need</p>
-------	--------------------	---

Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

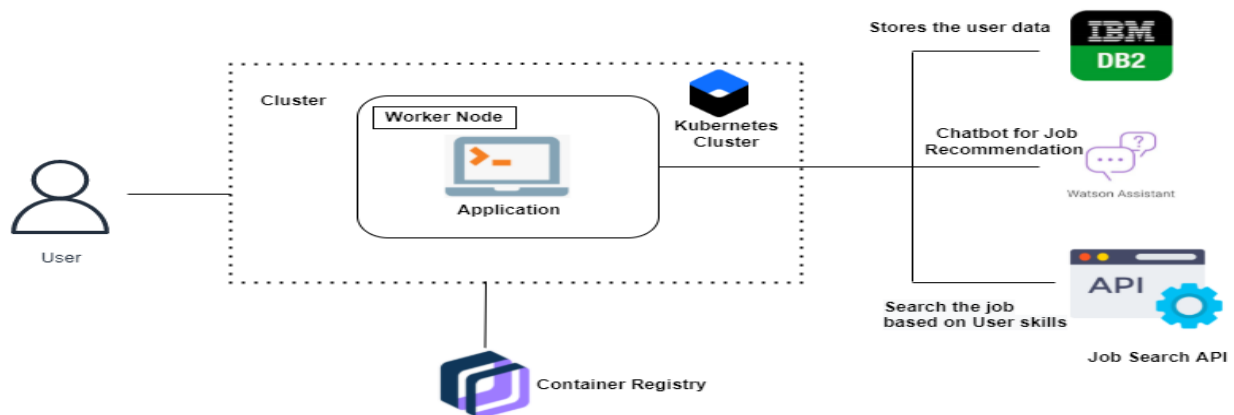


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	The user interacts with web application frontend, chatter bot, customer care, API, UI/UX	HTML, CSS, JavaScript,bootstrap, fontawesome,
2.	Application Logic-1	The application works with Python's flask framework includes inbuilt web libraries. All the backend logics are handled using flask. The request/ response are handled using Flask's WSGI.	Python - Flask Framework
3.	Application Logic-2	The user can interact using chatter bot which user websocket and end to end encryption	IBM Watson Assistant
4.	Application Logic-3	It is a Hybrid web application, which	Cloud service

		uses both public and private cloud service	
5.	Database	Due to application complexity and uses, relational databases are used. So database management system can be handled easily	MySQL, NoSQL, Postgresql etc.
6.	Cloud Database	On production online cloud database service is used, so it can be accessed through HTTP protocol	IBM DB2, IBM Cloudant etc.
7.	File Storage	All the required files that are saved in the web application are uploaded in an online file storage cloud service. So it can be accessed easily	IBM Block Storage or Other Storage Service or Local Filesystem
8.	External API-1	External Job API are used to provide job suggestions for the user in the application	Rapidapi
9.	External API-2	The identity of an individual can be	Aadhar API, etc.

		referred using external API provided by government	
10.	Machine Learning Model	ML is used in chatterbot to provide a better experience to the user.	Neural Machine Translation, communication etc.
11.	Infrastructure (Server / Cloud)	The application is used to run on a local server during the development phase. And it is deployed in online server during production	Local, Cloud Foundry, Kubernetes, Docker etc.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	This application uses python's Flask framework for development	Python language
2.	Security Implementations	Security, encryption are provided by flask security package such as session based authentication, password hashing, HTTP authentication , Token based authentication, JWT token authentication	Flask-security package
3.	Scalable Architecture	The python architecture is much scalable, so it can be developed further according to the needs in future	Flask Framework
4.	Availability	This application is available from anywhere at anytime independent of IP, due to it's server	IBM server
5.	Performance	This application uses good development architecture for performance, it is need to be developed in a	Redis

S.No	Characteristics	Description	Technology
		way to have a good user experience such as using cookies, sessions, cache	

MILE STONE

Product Backlog, Sprint Schedule, and Estimation

TITLE	DESCRIPTION	DATE	TEAM MEMBERS
Literature survey	In this activity you are expected to gather/collect the relevant information on project usecase, refer the existing solutions, technical papers, research publications etc.	3 sep 2022	Sainath, Adith, Adhithya, Karthick, Lokesh Kumar
Prepare empathy map	In this activity you	10 sep 2022	Sainath,

	are expected to prepare the empathy map canvas to capture the user Pains & Gains, Prepare list of problem statements.		Adith, Adhithya , Karthick, Lokesh Kumar
--	---	--	---

Ideation	In this activity you are expected to list the ideas (at least 4 per each team member) by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance.	17 sep 2022	Sainath , Adith, Adhithya a, Karthic k, Lokesh Kumar
Proposed solution	In this activity you are expected to prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc.	24 sep 2022	Sainath , Adith, Adhithya a, Karthic k, Lokesh Kumar

Problem solution fit	In this activity you are expected to prepare problem - solution fit document and submit for review.	26 sep 2022	Sainath , Adith, Adhithy a, Karthic k, Lokesh Kumar
Solution Architecture	In this activity you are expected to prepare solution architecture document and submit for review.	1 Oct 2022	Sainath , Adith, Adhithy a, Karthic k, Lokesh Kumar
Customer journey	Prepare the customer journey maps to understand the user interactions & experiences with the application (entry to exit).	3 Oct 2022	Sainath , Adith, Adhithy a, Karthic k, Lokesh Kumar
functional requirement	In this activity you are expected to prepare the functional requirement document.	8 Oct 2022	Sainath , Adith, Adhithy a, Karthic k, Lokesh Kumar

Data flow diagram	In this activity you are expected to prepare the data flow diagrams and submit for review.	10 Oct 2022	Sainath , Adith, Adhithya, Karthick, Lokesh Kumar
Technology Architecture	In this activity you are expected to draw the technology architecture diagram.	15 Oct 2022	Sainath , Adith, Adhithya, Karthick, Lokesh Kumar
Prepare Milestone	In this activity you are expected to prepare the milestones & activity list of the project.	17 Oct 2022	Sainath , Adith, Adhithya, Karthick, Lokesh Kumar
Sprint Delivery plan	In this activity you are expected to prepare the sprint delivery plan.	22 Oct 2022	Sainath , Adith, Adhithya, Karthick, Lokesh Kumar

Sprint 1	In this activity you are expected to develop & submit the developed code by testing it.	29 Oct 2022	Sainath , Adith, Adhithy a, Karthic k, Lokesh Kumar
Sprint 2	In this activity you are expected to develop & submit the developed code by testing it.	5 Nov 2022	Sainath , Adith, Adhithy a, Karthic k, Lokesh Kumar
Sprint 3	In this activity you are expected to develop & submit the developed code by testing it.	12 Nov 2022	Sainath , Adith, Adhithy a, Karthic k, Lokesh Kumar
Sprint 4	In this activity you are expected to develop & submit the developed code by testing it.	19 Nov 2022	Sainath , Adith, Adhithy a, Karthic k, Lokesh Kumar

SPRINT DELIVERY PLAN

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	5 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	22	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	23	9 Nov 2022

User Type	Functional Requir	User Story	User Story / Task	Acceptance criteria	Priority	Release

	ement (Epic)	Num ber				s e
Customer (Mobile user)	Registr ation	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	S pr in t- 1
Customer (Mobile user)	User registrat ion confirm ation	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	S pr in t- 1
Customer (Mobile user)	User registrat ion through third party web applicati on	USN -3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	S pr in t- 2
Customer (Mobile user)	User registrat ion through third party	USN -4	As a user, I can register for the application through Gmail	with google application	Medi um	S pr in t- 1
Customer (Mobile user)	Login	USN -5	As a user, I can log into the application by entering email & password	login system using flask	High	S pr in

						t-1
Customer (Mobile user)	Dashboard	USN -6	The activities and records of the user are rendered in dashboard	User can read, delete his/her record	High	Print-1
Customer (Web user)	Job suggestion	USN -7	The jobs are suggested as per the skills mentioned by the user in profile	from profile page	Low	Print-1
Customer Care Executive	Support for customers	USN -8	Doubts and suggestions can be clarified using chatbot and customer care number	Can used by user	High	Print-2

Administrator	Superuser	USN -9	All functionalities are managed by superuser called as Administrator	can accessed by specific user	High	Print-2
---------------	-----------	--------	--	-------------------------------	------	---------

CODING AND SOLUTION

accounts/forms.py

```

from django.contrib.auth.forms import UserCreationForm, UserChangeForm
from django.contrib.auth import get_user_model

from .models import Profile
from django.forms import ModelForm, CharField, IntegerField, ChoiceField,
NumberInput

CustomUser = get_user_model()
class CustomUserCreationForm(UserCreationForm):

    class Meta:
        model = CustomUser
        fields = ("username", "email")

class CustomUserChangeForm(UserChangeForm):

    class Meta:
        model = CustomUser
        fields = ("username", "email")

class ProfileForm(ModelForm):
    # specialization = CharField(widgets =
CharField(attrs={'placeholder': 'Enter your Field of
specialization', 'maxlength':50})),
    # interested_domain = CharField(widgets = CharField(
attrs={'placeholder': 'enter your interested domain. ex: Web
development', 'maxlength':300})),
    # skills = CharField(widgets = CharField(attrs={'placeholder':
'Enter your skills example, Ex: python', 'maxlength':200})),
    # address1 = CharField(widgets = CharField(attrs={'placeholder':
'address1', 'maxlength':400})),
    # address2 = CharField(widgets = CharField(attrs={'placeholder':
'address2', 'maxlength':400})),
    # country = CharField(widgets = CharField(attrs={'placeholder':
'Country', 'maxlength':100})),
    # pincode = NumberInput(widgets = NumberInput(
attrs={'placeholder': 'Enter the pincode', 'maxlength': 6 })),

```

```

        # gender = ChoiceField()
        class Meta:
            model = Profile
            fields = ['specialization', 'interested_domains', 'skills',
'address1', 'address2', 'country', 'pincode', 'gender']

```

accounts/models.py

```

import uuid
from django.db import models
from django.contrib.auth.models import AbstractUser
from django.urls import reverse
from django.conf import settings

class CustomUser(AbstractUser):
    id = models.UUIDField(primary_key=True, default=uuid.uuid4,
editable=False)

    def __str__(self):
        return f"{self.username}"

class Profile(models.Model):
    class Gender(models.TextChoices):
        male = "Male",
        female = "Female"

    id = models.UUIDField(primary_key=True, default=uuid.uuid4,
editable=False)
    user = models.ForeignKey(settings.AUTH_USER_MODEL,
related_name='profile_user', on_delete=models.CASCADE)
    specialization = models.CharField(max_length = 50, blank=False)
    interested_domains = models.CharField(max_length=300, blank=False)
    skills = models.CharField(max_length=200, blank=False)
    address1 = models.CharField(max_length=400, blank=False)

```

```

address2 = models.CharField(max_length=400, blank=False)
country = models.CharField(max_length=100)
pincode = models.IntegerField(blank=False, null=True) #fix max value
gender = models.CharField(max_length=20, choices=Gender.choices)


def __str__(self):
    return f"{self.user}"

```

accounts/signals.py

```

from allauth.account.signals import user_signed_up
from django.dispatch import receiver
from django.contrib.auth import get_user_model
from django.shortcuts import get_object_or_404
from .models import Profile

CustomUser = get_user_model()

@receiver(user_signed_up)
def user_signed_up_set_profile(request, user, **kwargs):
    current_user = CustomUser.objects.get(email=user.email)
    object = Profile(user=current_user)
    object.save()

```

accounts/admin.py

```

from django.contrib import admin
from django.contrib.auth.admin import UserAdmin
from django.contrib.auth import get_user_model
from .forms import CustomUserChangeForm, CustomUserCreationForm
from .models import Profile

```

```

CustomUser = get_user_model()
class CustomUserAdmin(UserAdmin):
    add_form = CustomUserCreationForm
    form = CustomUserChangeForm
    model = CustomUser
    list_display = ["email", "username",]

admin.site.register(CustomUser, CustomUserAdmin)
admin.site.register(Profile)

```

main_application.py/models.py

```

import uuid
from django.db import models
from django.contrib.auth.models import AbstractUser
from django.urls import reverse
from django.conf import settings
from accounts.models import Profile

class JobApplied(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid.uuid4,
editable=False)
    user = models.ForeignKey(Profile, related_name='job_profile',
on_delete=models.CASCADE)
    title = models.CharField(max_length=300, blank=True)
    location = models.CharField(max_length=500, blank=True)
    company_name = models.CharField(max_length=500, blank=True)

    def __str__(self):
        return f"{self.user}"

```

main_application/views.py

```
from django.shortcuts import render, redirect
from django.contrib.auth import get_user_model
from django.shortcuts import get_object_or_404
from django.contrib.auth.decorators import login_required
from django.http import HttpResponseRedirect
from accounts.forms import ProfileForm
from accounts.models import Profile
import requests
import json
from .models import JobApplied

CustomUser = get_user_model()
@login_required(login_url='account_login')
def home(request):

    job_list = []
    url = requests.get('https://arbeitnow.com/api/job-board-api')
    response = url.text
    json_data = json.loads(response)
    for i in range(0,40):

        a = json_data["data"][i]["company_name"]
        b = json_data["data"][i]["title"]
        c = json_data["data"][i]["location"]
        job_list.append([a,b,c])

    id = request.user.id
    current_user = get_object_or_404(CustomUser, id=id)
    profile = Profile.objects.get(user=current_user)
    if request.method == "POST":
        # form = JobApplied(request.POST)
        data_submitted = request.POST.dict()
        object = JobApplied(user=profile, title=data_submitted["title"],
location=data_submitted["location"], company_name=
data_submitted["company_name"])
        object.save()
```



```

        print(data_submitted)

    context = {
        "job_list":job_list,

    }
    return render(request, 'home.html', context)

@login_required(login_url='account_login')
def profile(request, id):
    current_user = get_object_or_404(CustomUser, id=id)
    profile = Profile.objects.get(user=current_user)
    job_applied = JobApplied.objects.filter(user=profile).all()
    if current_user != request.user:
        return HttpResponseBadRequest('error 404')
    else:

        form = ProfileForm(instance=profile)
        if request.method == 'POST':
            form = ProfileForm(request.POST, instance=profile)
            if form.is_valid():

                save_form = form.save(commit=False)
                save_form.user = request.user
                form.save()
                print("es")
                return redirect('profile', id=str(request.user.id))
    context = {
        'form':form,
        'job_applied': job_applied
    }
    return render(request, 'profile.html', context)

```

Requirements.txt

```
asgiref==3.5.2
certifi==2022.9.24
cffi==1.15.1
charset-normalizer==2.1.1
cryptography==38.0.3
defusedxml==0.7.1
Django==4.1.3
django-allauth==0.51.0
django-crispy-forms==1.14.0
django-smtp-ssl==1.0
ibm-db==3.1.3
ibm-db-django==1.5.2.0
idna==3.4
oauthlib==3.2.2
pycparser==2.21
PyJWT==2.6.0
python3-openid==3.2.0
regex==2022.10.31
requests==2.28.1
requests-oauthlib==1.3.1
six==1.16.0
sqlparse==0.4.3
tzdata==2022.6
urllib3==1.26.12
```

RESULT

Home page

Jobfindo

Home Account Logout

Company name

airkom Druckluft GmbH

Title

VertriebsberaterIn (m/w/d)

Location

Wildau

submit

Company name

AUGUSTIN & PARTNER mbB Steuerberater Wirtschaftsprüfer


Title

Steuerberater mit Partneroption (m/w/d)

Account or profile page

Jobfindo

Home Account Logout Database



Specialization

cdwqdc

Job you have applied

Title: Rüsselsheim sucht nächste Java-Helden! (d/w/m)

location: Rüsselsheim

company name: NXT Hero GmbH

Title: Projektmanager*in (all genders) für die pharmazeutische Qualitätskontrolle | 492527

location: Frankfurt

company name: Hox Life Science GmbH

Login page

Job findo



LOGIN

E-mail address

Password

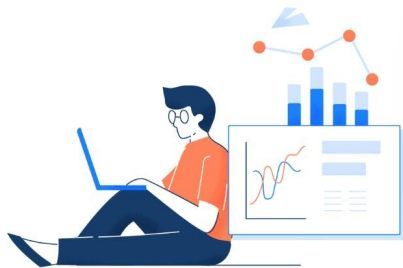
☐ Remember Me

[Sign In](#)

[Forgot Password?](#)

Signup page

Job findo



SIGN UP

E-mail address

Password

Password (again)

[Sign Up](#)

Already have an account? Then please [sign in](#).

Conclusion

Hence by required requirements the software have been successfully deveopled

