An Advanced Learner Project Report

ON

“Job Predictor for Differently Abled”

Submitted to

UNIVERSITY OF MUMBAI

In Partial Fulfilment of the Requirement for the Award of

BACHELOR’S DEGREE IN

INFORMATION TECHNOLOGY

ENGINEERING

BY

Name 1 (BE). UID:

Name 2 (BE). UID:

Name 3 (BE) UID:

**UNDER THE GUIDANCE OF**

**PROF. Ankit Anand**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**MGM COLLEGE OF ENGINEERING**

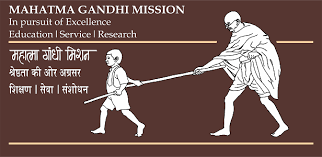
**KAMOTE, NAVI MUMBAI - 410209**

**2020-2021**

MGM COLLEGE of ENGINEERING

Department of Information Technology

KAMOTE, NAVI MUMBAI 410209



CERTIFICATE

This is to certify that the project entitled

“Job Predictor for Differently Abled "

Submitted by

Name 1(BE-IT) UID:

Name 2(BE-IT) UID:

Name 3(BE-IT) UID:

Is a record of bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering ( Information Technology Engineering) at MGM College of Engineering, Navi Mumbai under the University of Mumbai. This work is done during year 2020-2021, under our guidance.

Date: / /

(Prof. Ankit Anand) (Prof. K.V.Raman)

(Project Guide) (HOD, IT Department)

Acknowledgements

We are profoundly grateful to Prof.Ankit Anand for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

We would like to express deepest appreciation towards Dr. S.K.Narayan Khedkar, Principal, MGM College, Prof.K.V.Raman, Head of Department of Information Technology Engineering and Prof. Ankit Anand , Project Coordinator whose invaluable guidance supported us in completing this project.

At last we must express our sincere heartfelt gratitude to all the staff members of Computer Engineering Department who helped me directly or indirectly during this course of work.

Name 1 (BE)

Name 2 (BE)

Name 3 (BE)

ABSTRACT

Currently, there are various websites available in the internet that are providing jobs for normal as well differently abled people. But the limitation is that only some percent of differently abled gets into a job and majority of them are jobless. We would like to propose a system which would concentrate on their strengths and predict the suitable job for them.

We have divided the project into 3 modules

* Job Prediction
* Job Recommendation
* Chatbot

Contents

|  |  |  |
| --- | --- | --- |
| No | Topic | Page |
| 1 | Software Requirement and specification   * 1. Product Perspective ………………………..      1. Specific Requirement………………... | 6  6 |
| 2 | Project Plan  2.1 Designing Job Predictor……………………..  2.2 Designing a Job Recommender ……………...  2.3 Building Chatbot……………………………..  2.4 Storing to firebase…………………………… | 7  8  9  10  11 |
| 3 | Architecture and Flowchart | 12 |
| 4 | Screenshots of project  Fig (a) Home page………………………………..  Fig (b) Pan Card Scanned Image…………….  Fig (c) Driving Licence Scanned Image.........  Fig (d) Driving Licence Detected Image......... | 13  14  15  16 |
| 5 | Conclusion And Future scope  5.1 Conclusion…………………………………  5.2 Future scope………………………………. | 17  17 |
|  | References | 18 |
|  |  |  |
|  |  |  |

Chapter 1

System Analysis

* 1. Existing System

The current system doesn’t focus on providing jobs for differently abled people. The current system provides job for normal people and so many disabled people are jobless

* 1. Proposed System

The proposed system would predict as well as recommend the suitable jobs for differently abled people so that they would get their desired jobs easily and efficiently.

Chapter 2

Software Requirements

2.1 **Product Perspective**

The proposed system is a Web App built using python flask framework. The web app could be connected easily in any job site where the service is needed and then authorized person would be able to view the available jobs predicted for them.

2.1.1 **Specific Requirements**

The whole project is developed using flask Framework which uses python programming language and deployed using Heroku Cloud . So the overall software requirements of the project are:-

* flask version 1.1.2
* Pycharm or any other IDE
* Windows 7 or another Operating system
* Heroku

As an end product a web app will be developed by us which would then be connected with any job site to avail our service.

Chapter 3

Project Planning

The overall aspect of the project work has been to develop a proper and simple management of the data using web app.

A detailed study of internet programming, software development and cloud was conducted as an initial stage.

The study of python, CSS3, Bootstrap4.5, HTML5 have also helped in our development work

The major plan that will be executed for the purpose of designing are

* Developing the model (Job predictor and Recommender System)
* Designing the Job Predictor Web App
* Connecting Chatbot
* Firebase (Database and Authentication)
* Deploying the App on Heroku

3.1 **DEVELOPING THE MODEL**

We are developing a Random Forest Classifier and Kmeans Clustering model to predict and recommend jobs respectively based on skills, salary range and type of disabilities which the user enters into the web app. Model is trained using Sklearn and saved in compressed form (pickle) for ease of access and space reduction.

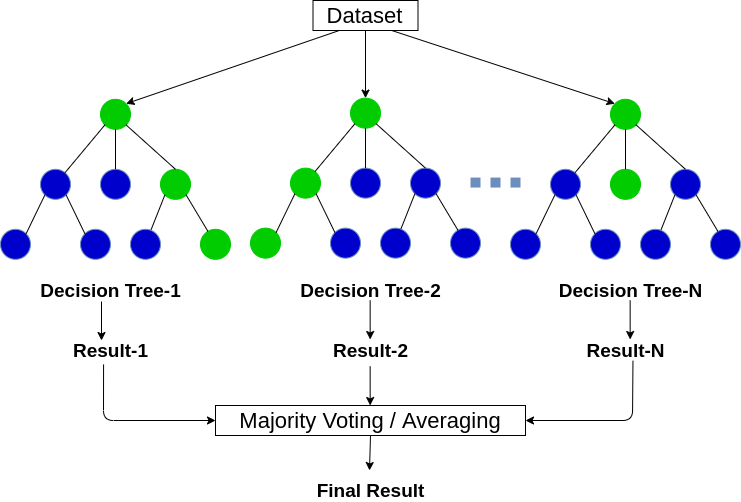


Fig 1 Random Forest Model

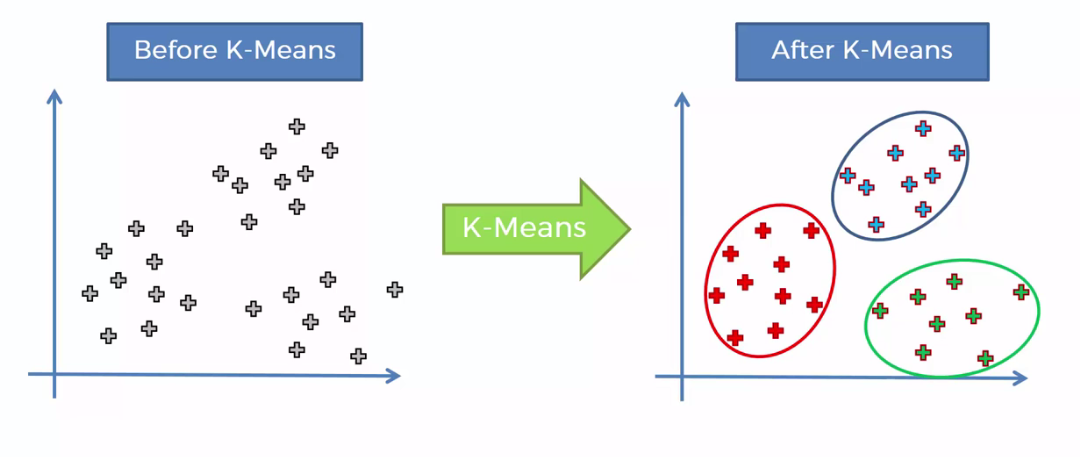


Fig 2 Kmeans Clustering Model (For Recommending Jobs)

3.2 **DESIGNING THE WEB APP**

The App would be created using flask framework. The Prediction model would be loaded from pickle file. This model would receive the required inputs from user and predict the most suitable jobs for disabled people. The Recommender would also work in similar way but with a clustering model.

We would recommend top five jobs for the disabled people.

3.3 **DEVELOPING CHATBOT**

We’ve used Google Dialogflow to make the chatbot. Kommunicate was used to integrate the chatbot with our web app. Google Dialogflow provides a nice UI to design and test the chatbot. Once we are satisfied kommunicate can be used to integrate the bot. The bot answers most of the users queries regarding working of our web app.

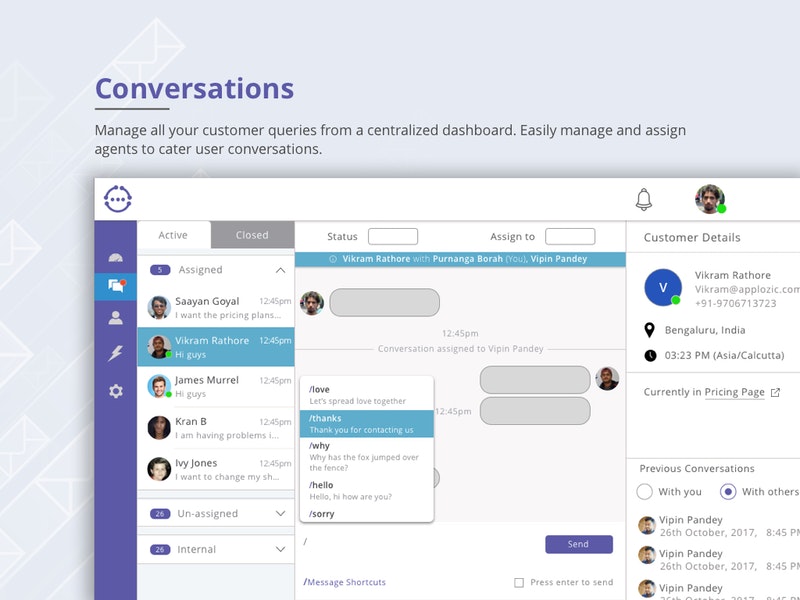


Fig 3 Kommunicate Dashboard

* 1. **FIREBASE (DBaaS) CONNECTIVITY**

3.4.1. Firebase for Authentication

Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more.

Firebase Authentication integrates tightly with other Firebase services, and it leverages industry standards like OAuth 2.0 and OpenID Connect, so it can be easily integrated with your custom backend.

We’ve used firebase authentication service which is a DBaaS (Data Base as a Service) for our user login and register process. The email and password of user is stored securely in firebase which would be used for authenticating the user.

* + 1. Firebase as Realtime Database

The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in realtime to every connected client. When you build cross-platform apps with our iOS, Android, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data.

We’ve used realtime database to store the profile of users as well as their their resume.



Fig 4 Firebase Authentication

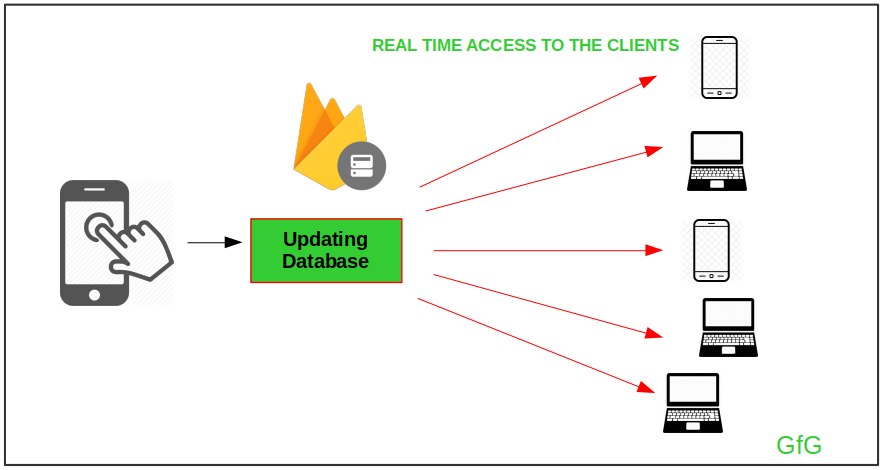


Fig 5 Firebase Realtime Database

* 1. **Deployment in Heroku**

Heroku makes it easy to deploy and scale python apps. Whether its Django or flask heroku can deploy them easily. It is a PaaS (Platform as a Service) provided by Salesforce for free to build and deploy app. It helps students to get knowledge of deployment in cloud. We’ve also used Heroku to deploy our web app which would predict and recommend job for disabled people.

The link of our project is: -

<https://job-pred.herokuapp.com>

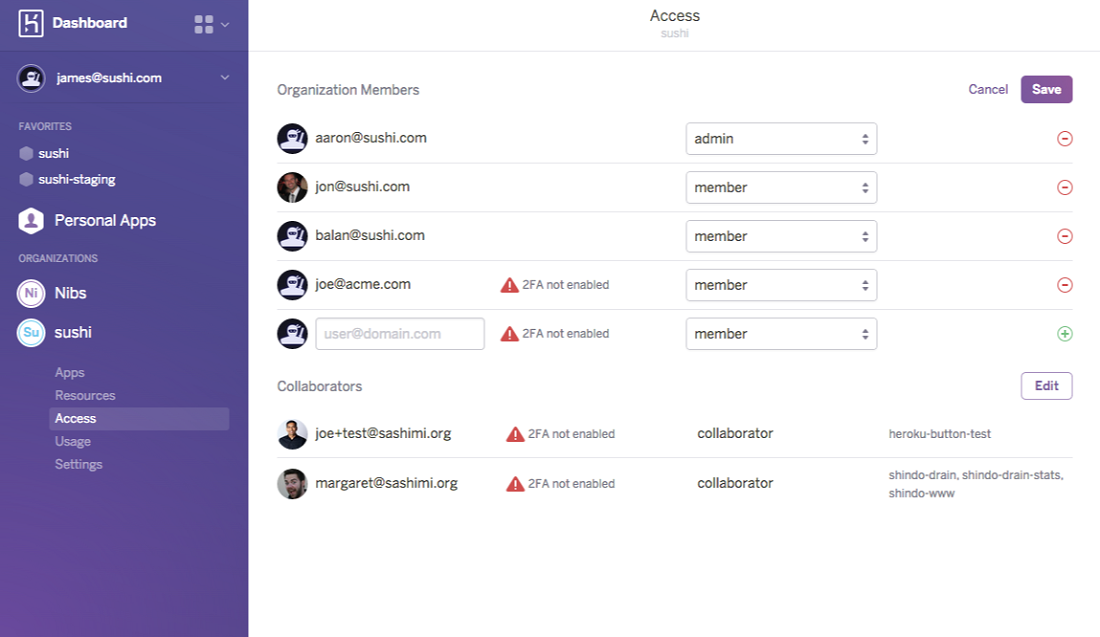
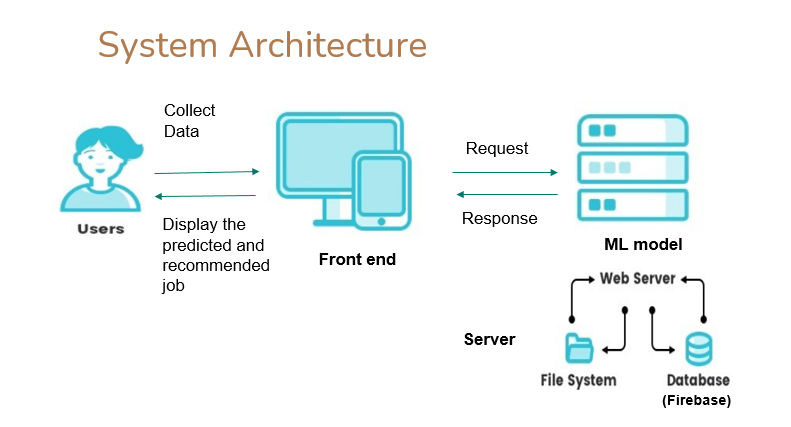


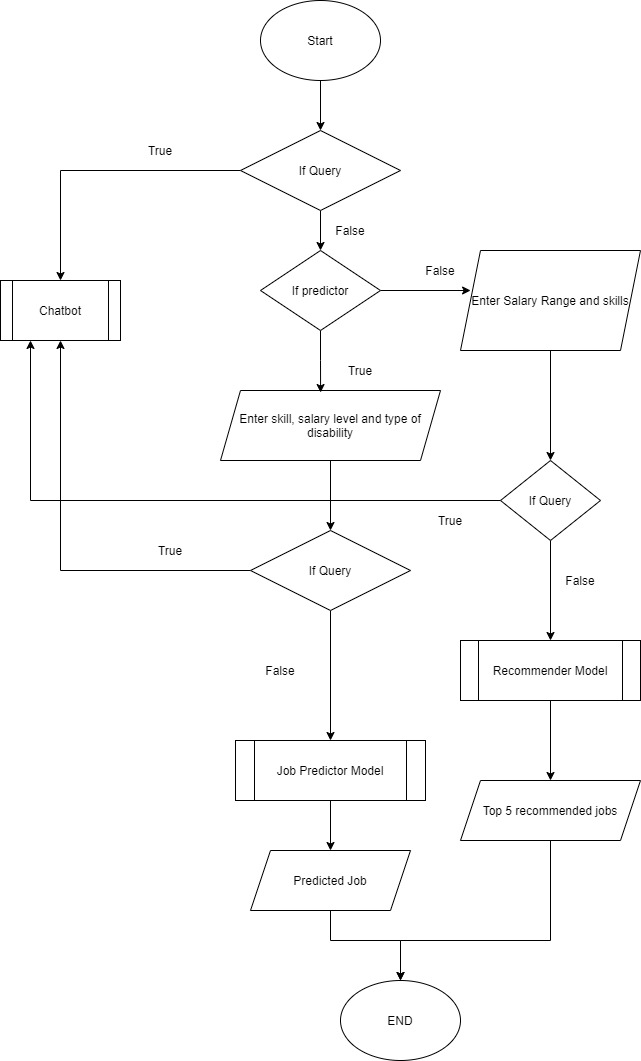
Fig 6 Heroku Dashboard

**Chapter 4**

**ARCHITECTURE**

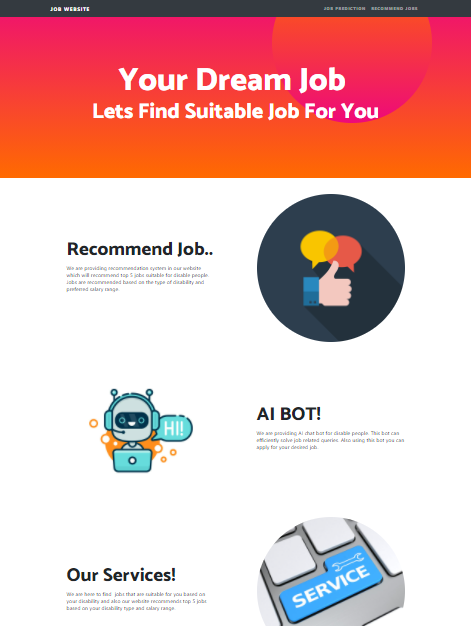
****

**FLOW CHART**

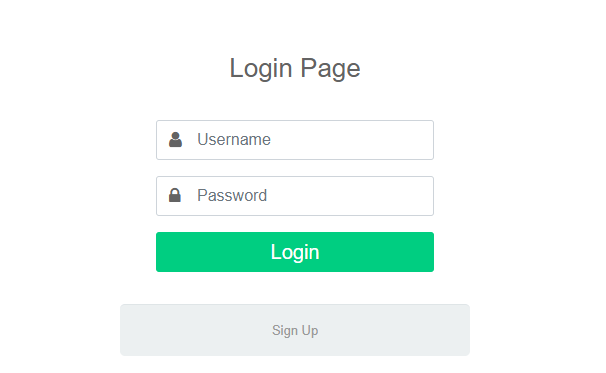
****

Chapter 5

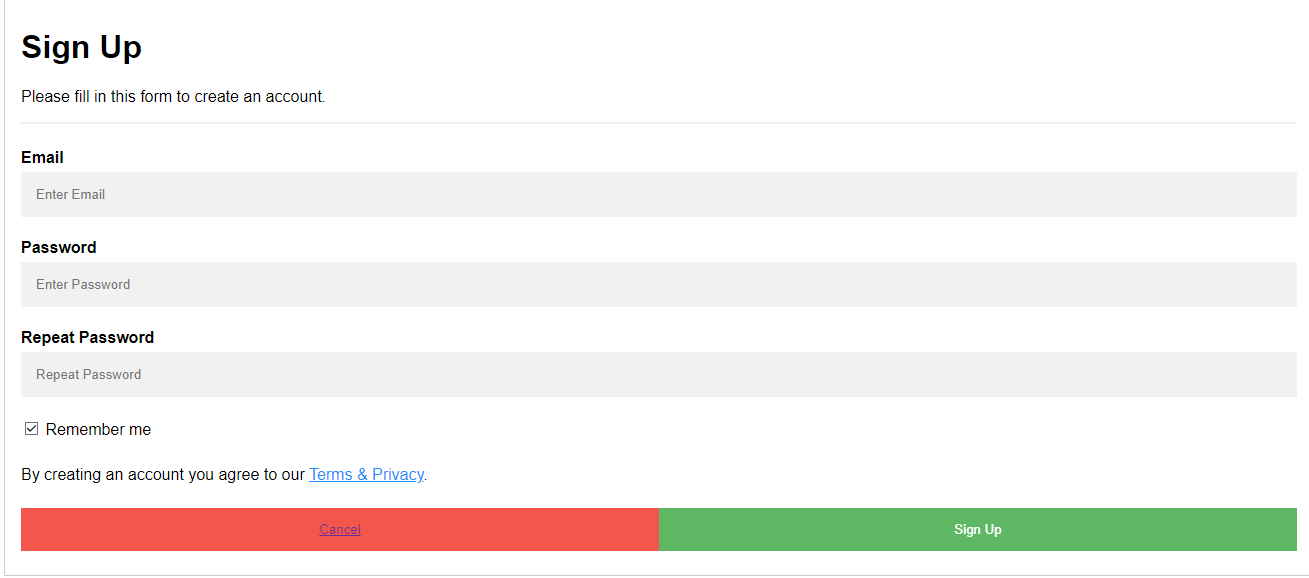
Snippets



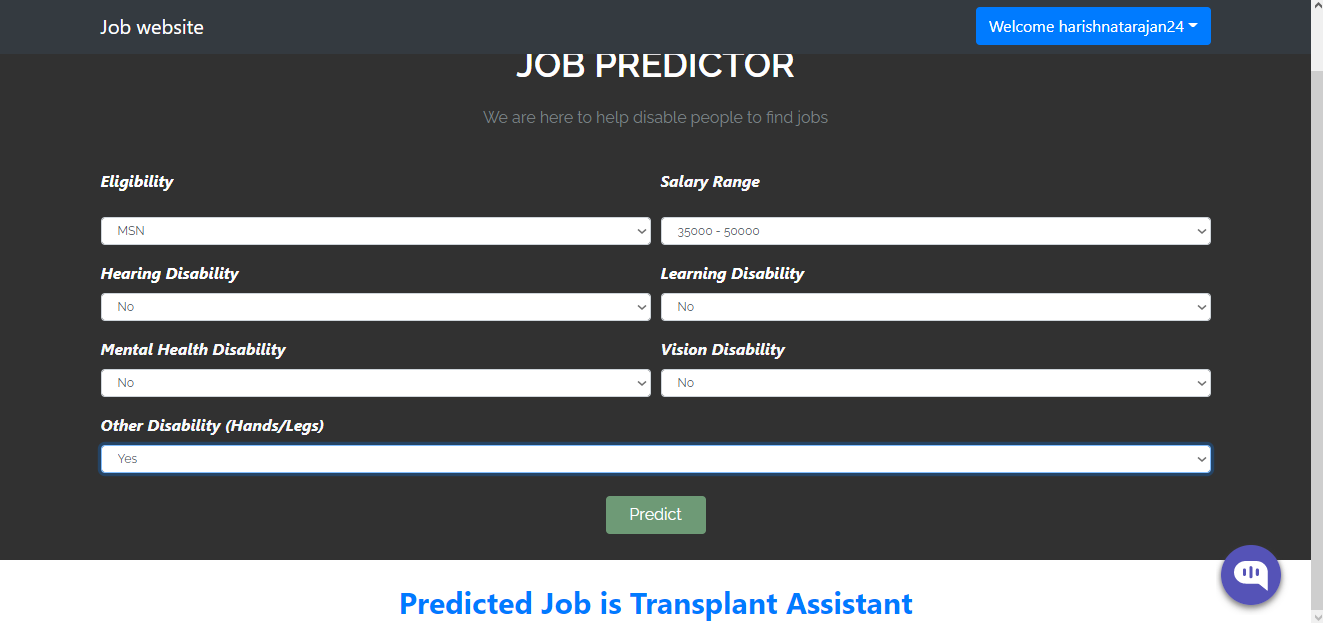
Home Screen



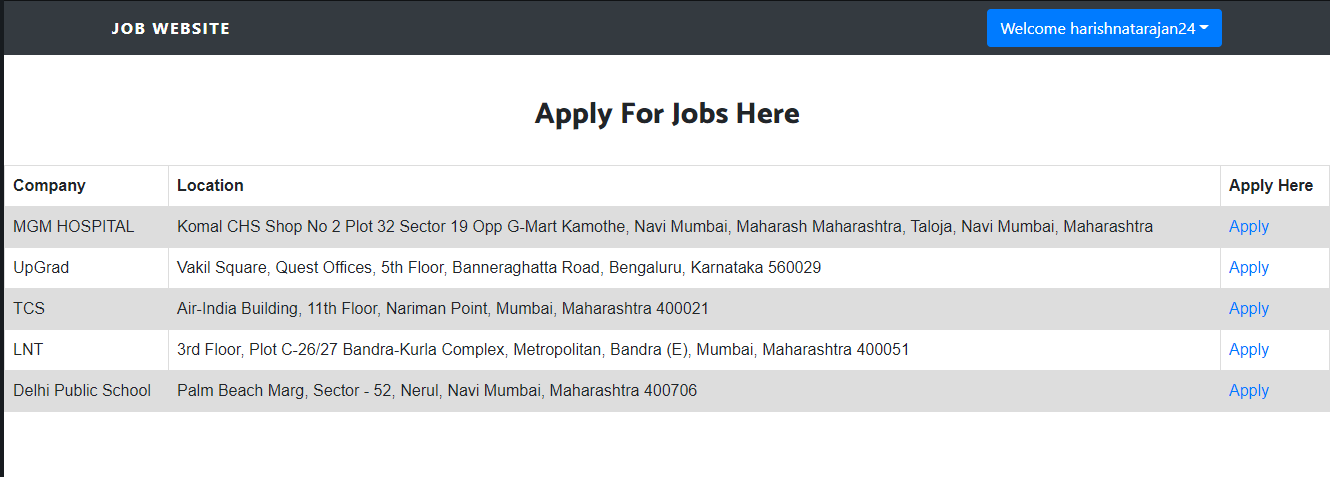
Login



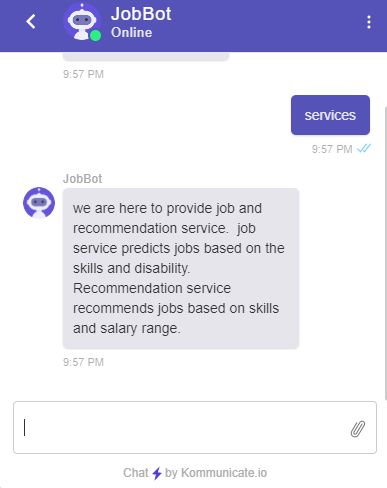
Register



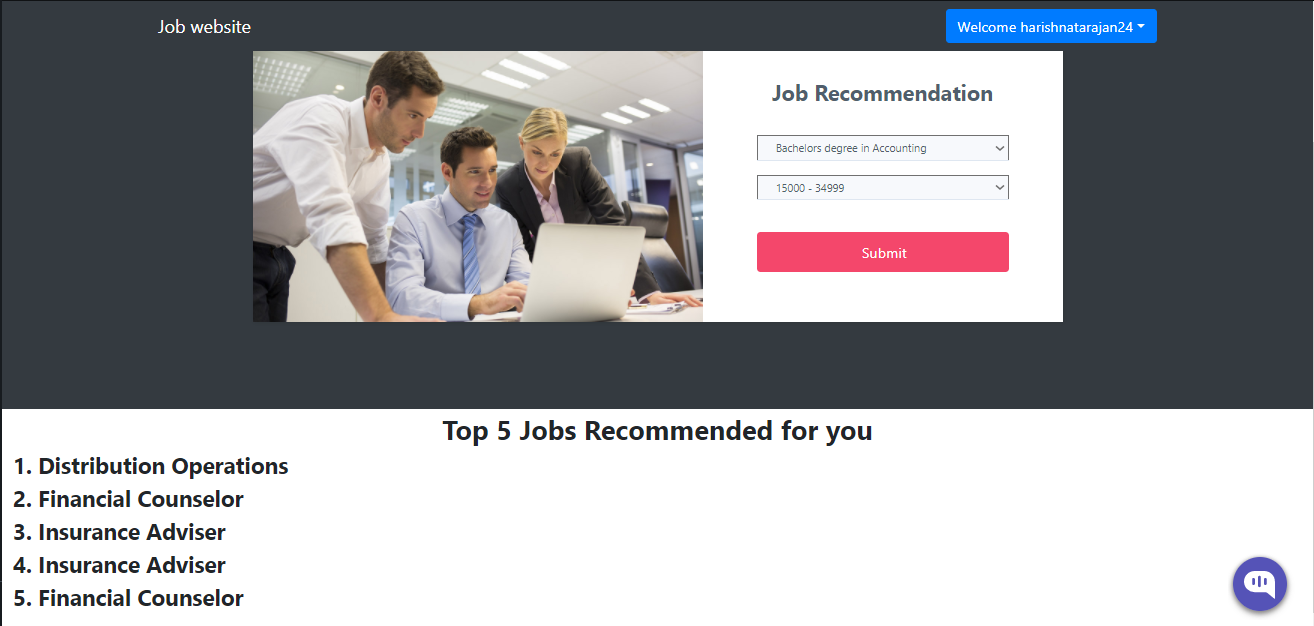
Job Prediction



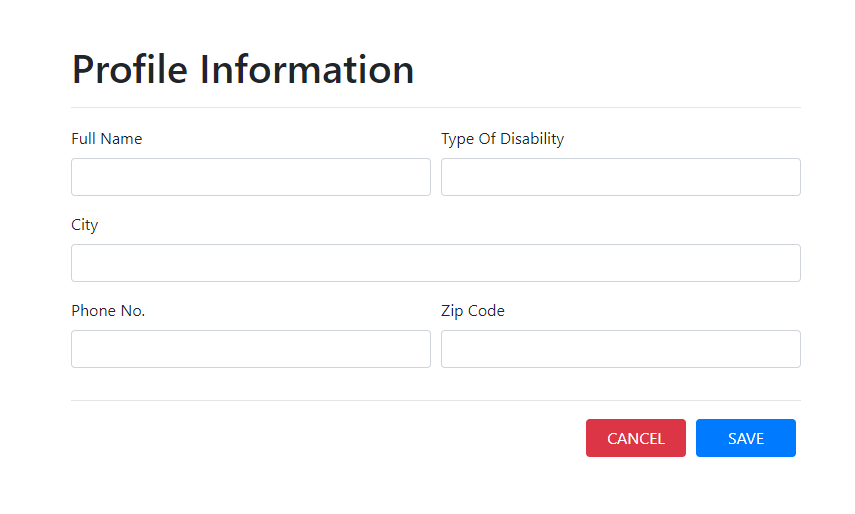
Apply for job



Chatbot



Recommend Job



Profile



Resume Uploader

Chapter 6

Code Snippets

6.1. App code

from flask import Flask, render\_template, request, redirect, session

import pandas as pd

import ast

import PyPDF2

from firebase import authentication, db

import pickle

app = Flask(\_\_name\_\_)

app.secret\_key = "ReallySecret"

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

def poster():

return render\_template("poster.html")

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

def login():

return render\_template("login.html")

@app.route("/info")

def info():

return render\_template("profile.html")

@app.route("/suggestion")

def suggestion():

if "user" in session:

return render\_template("suggestion.html", user\_name=session["user"])

else:

return render\_template("login.html", login\_fail="You have not logged in")

@app.route("/resume")

def resume():

return render\_template("resume\_upload.html")

@app.route('/signup')

def signup():

return render\_template("signup.html")

@app.route('/home', methods=['GET', 'POST'])

def home():

if "user" in session:

eli\_list, sal\_list, le\_eli, le\_sal, model = db.load\_model()

leng\_eli = len(list(eli\_list))

len\_sal = len(list(sal\_list))

eli = request.form.get('eligibility')

sal = request.form.get('salary')

if request.method == "POST":

hearing = request.form.get('hearing')

vision = request.form.get('vision')

learning\_disability = request.form.get('learning\_disability')

mental\_health = request.form.get('mental\_health')

other = request.form.get('other')

test\_data = pd.DataFrame(

[

{"Hearing": int(hearing),

"Learning": int(learning\_disability),

"Mental Health": int(mental\_health),

"Other (Hand/Leg)": int(other),

"Vision": int(vision),

"Eligibility": eli,

'Salary': sal

}

]

)

test\_data['Salary'] = le\_sal.transform(test\_data['Salary'])

test\_data['Eligibility'] = le\_eli.transform(test\_data['Eligibility'])

pred = model.predict(test\_data)[0]

return render\_template('home.html', leng\_eli=leng\_eli,

list\_eli=list(eli\_list), len\_sal=len\_sal,

list\_sal=list(sal\_list), prediction=f'Predicted Job is {pred}',

user\_name=session["user"]

)

return render\_template('home.html',

leng\_eli=leng\_eli,

list\_eli=list(eli\_list),

len\_sal=len\_sal,

list\_sal=list(sal\_list),

user\_name=session["user"]

)

else:

return render\_template("login.html", login\_fail="You have not logged in")

@app.route('/recommend', methods=['GET', 'POST'])

def recommend():

if "user" in session:

list\_eli, sal\_list, le\_eli, le\_sal, model = db.load\_model()

leng\_eli = len(list(list\_eli))

len\_sal = len(list(sal\_list))

reccomend = pickle.load(open('./models/recommend\_model.pkl', 'rb'))

if request.method == "POST":

eli = request.form.get('eligibility')

sal = request.form.get('salary')

test\_data = pd.DataFrame(

[

{

"Eligibility": eli,

'Salary': sal

}

]

)

test\_data['Salary'] = le\_sal.transform(test\_data['Salary'])

test\_data['Eligibility'] = le\_eli.transform(test\_data['Eligibility'])

pred = reccomend.predict(test\_data)[0]

job\_data = pd.read\_csv('./models/Recommended.csv')

comp\_list = ast.literal\_eval(job\_data.T[pred]['Recommended Job'])

prediction = comp\_list[0:5]

return render\_template('recommend.html', len\_sal=len\_sal,

len\_eli=leng\_eli, list\_eli=list\_eli,

list\_sal=sal\_list, prediction=prediction,

user\_name=session["user"]

)

return render\_template('recommend.html', len\_sal=len\_sal,

len\_eli=leng\_eli, list\_eli=list\_eli,

list\_sal=sal\_list, user\_name=session["user"]

)

else:

return render\_template("login.html", login\_fail="You have not logged in")

@app.route("/login\_up", methods=['POST'])

def login\_up():

eli\_list, sal\_list, le\_eli, le\_sal, model = db.load\_model()

leng\_eli = len(list(eli\_list))

len\_sal = len(list(sal\_list))

username = request.form["username"]

password = request.form["password"]

lo = authentication.login1(username, password)

if lo:

user1 = authentication.userNameExtract(username)

session["user"] = user1

return render\_template('home.html',

leng\_eli=leng\_eli,

list\_eli=list(eli\_list),

len\_sal=len\_sal,

list\_sal=list(sal\_list),

user\_name=session["user"])

else:

return render\_template("login.html", login\_fail="You have entered wrong email or password")

@app.route('/sign\_up', methods=['POST'])

def sign\_up():

email = request.form["email"]

password = request.form["psw"]

con\_password = request.form["psw-repeat"]

if password == con\_password and len(password) >= 6:

sign\_up1 = authentication.signup1(email, password)

if sign\_up1:

return render\_template("login.html")

else:

return render\_template("signup.html", signup\_fail="Email already exists")

elif password == con\_password:

return render\_template("signup.html", signup\_fail="Password must be at least 6 characters")

else:

return render\_template("signup.html", pass\_fail="Passwords not matching ")

@app.route("/logout", methods=["GET", "POST"])

def logout():

session.pop("user", None)

return render\_template("login.html")

@app.route("/profile\_up", methods=["POST"])

def profile\_up():

fullname = request.form["fullname"]

tod = request.form["tod"]

city = request.form["city"]

tel = request.form["tel"]

zip1 = request.form["zip"]

dict1 = {"fullname": fullname, "tod": tod, "city": city, "tel": tel, "zip": zip1}

db.insertDetails(dict1)

return render\_template("poster.html")

@app.route('/resume\_up', methods=['POST'])

def resume\_up():

if 'file' not in request.files:

return redirect(request.url)

file = request.files.get('file')

if not file:

return render\_template('resume\_upload.html')

reader = PyPDF2.PdfFileReader(file)

size = reader.numPages

lx = []

for i in range(0, size):

page = reader.getPage(i)

pdfData = page.extractText()

lx.append(pdfData)

page\_end = f"End of page {i + 1}"

lx.append("\n")

lx.append(f"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*{page\_end} \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

lx.append("\n")

result = "".join(lx)

del lx

db.insertPdf(result)

return render\_template("poster.html")

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

6.2. Firebase

6.2.1. Authentication

import pyrebase

config = {

"apiKey": "AdsdfSfdff",

"authDomain": "auth12demo-aghhf5f1.firebaseapp.com",

"databaseURL": "https://databaseName.fire123baseio100.com",

"storageBucket": "authdemo-af5f1.appspot354543.com"

}

firebase = pyrebase.initialize\_app(config)

auth = firebase.auth()

# lx = []

def login1(email, password):

try:

login = auth.sign\_in\_with\_email\_and\_password(email, password)

return True

except:

# print("Invalid email or password")

return False

def userNameExtract(username):

lx = []

for i in str(username):

if i == "@":

break

else:

lx.append(i)

result = "".join(lx)

del lx

return result

def signup1(email, password):

try:

user = auth.create\_user\_with\_email\_and\_password(email, password)

return True

except:

return False

6.2.2. DB (Database)

import pyrebase

import pickle

firebaseConfig = {

"apiKey": "AIzaSSdSfyD9f3HFSy87eIdAdVBa2LGeZxiVXsXnSlY",

"authDomain": "auth443545-b9834.firebaseapp.com",

"databaseURL": "https://auth54646-b9834-default-rtdb.firebaseio.com",

"projectId": "auth-b67889834",

"storageBucket": "auth-b67889834.appspot.com",

"messagingSenderId": "285887870819001",

"appId": "1:2858808198789001:web:a15af1c999b55b9cf39ce8",

"measurementId": "G-MTPKGHZLM9Z4TR"

}

firebase1 = pyrebase.initialize\_app(firebaseConfig)

db = firebase1.database()

def insertDetails(dict):

db.child("Details").push({"Details": dict})

def insertPdf(pdfData):

db.child("Pdf Data").push({"Resume Details": pdfData})

def load\_model():

le\_eli = pickle.load(open('./models/labelencoder\_eligibility.sv', 'rb'))

le\_sal = pickle.load(open('./models/labelencoder\_salary.sv', 'rb'))

eli\_list = pickle.load(open('./models/eligibility\_list.sv', 'rb'))

sal\_list = pickle.load(open('./models/Salary.sv', 'rb'))

model = pickle.load(open('./models/model.sv', 'rb'))

return eli\_list, sal\_list, le\_eli, le\_sal, model

chapter 7

Conclusion And Future Scope

7.1 Conclusion

The goal of the project was to predict and recommend jobs to differently abled people. By accessing this web app, disabled people can get to know for sure what jobs they will get.

7.2 Future Scope

Currently the model is trained using synthetic data obtained from different web sites. Since the memory was less the model was trained for short span of time. Also more data could be collected using Big Data and the model could be tuned further to improve it. We would also integrate application for jobs.

References

1. <https://flask.palletsprojects.com/en/1.1.x/>
2. <https://pythonhosted.org/PyPDF2/>
3. <https://github.com/thisbejim/Pyrebase>
4. <https://realpython.com/tutorials/flask/>
5. <https://bootstrapstudio.io/tutorials/>
6. <https://www.kommunicate.io/blog/integrate-bot-using-dialogflow-in-kommunicate/>
7. <https://cloud.google.com/dialogflow/docs>
8. <https://stackabuse.com/deploying-a-flask-application-to-heroku/>
9. <https://firebase.google.com/docs/auth>
10. <https://firebase.google.com/docs/database>