**Internship project on**

AI RESUME SCORE CARD

Submitted to

The Department of Computer Science and Engineering

Bachelor of Technology In Computer Science and Engineering

Eesha Devara 23R11A6759

G.Lalith Siddhardh 23R11A6764

J. Pranav Kumar 23R11A6768

Under the Guidance of

Adgaonker Shashank

Adgaon

Department of Computer Science and Engineering

**Geethanjali College of Engineering and Technology (UGC Autonomous)**

(Affiliated to J.N.T.U.H, Approved by AICTE, New Delhi, Accredited by NBA)

Cheeryal (V), Keesara (M), Medchal.Dist.-501 301.

**June-2025**

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Contents** | **Page No.** |
| 1. | Abstract | 3 |
| 2. | Introduction | 4 |
| 3. | Problem Statement | 5 |
| 4. | Solution | 6 |
| 5. | Program(index.html) | 7 |
| 6. | Program(App.py) | 15 |
| 7. | Program(Resume\_Utils) | 18 |
| 8. | Output | 19 |
| 9. | Algorithm | 20 |

**ABSTRACT:**

This project presents an AI-powered Resume Scorecard Generator designed to automate and enhance the resume evaluation process. The system is capable of analyzing unstructured resume data from various formats, including PDF, DOCX, and plain text, to identify critical entities such as skills, work experience, education, and certifications. By analyzing target job descriptions, the system establishes role-specific evaluation criteria and applies a configurable scoring algorithm to quantify alignment across key dimensions, including skill relevance, experience depth, qualifications, and document structure.

The implementation features a web-based interface that allows users to upload documents and receive comprehensive scorecards with actionable feedback. Utilizing technologies such as spaCy for semantic analysis and Flask/React for application architecture, the solution aims to reduce recruitment bias while providing job seekers with data-driven insights for resume optimization. Future enhancements may incorporate machine learning to adapt scoring models based on historical hiring patterns, positioning the tool as a scalable resource for both candidates and HR professionals seeking objective resume assessments.

**Key features include:**

# Here are the key features of an AI Resume Scorecard in bullet points:

# Resume Parsing – Extracts structured data (name, skills, education, etc.) from PDF, DOCX, or image files.

# Skill Matching – Compares resume skills with job description using NLP techniques like TF-IDF and cosine similarity.

# Scoring System – Provides an overall match score out of 100 based on relevance to the job role.

# AI-Powered Analysis – Analyzes grammar, tone, and quality of content using NLP/ML models.

# Section Completeness Check – Verifies presence of essential sections like Contact Info, Skills, Experience, Education, etc.

# Multi-format File Support – Supports PDF, DOCX, and image formats with OCR for scanned resumes.

# INTRODUCTION:

# In today’s competitive job market, employers are increasingly leveraging artificial intelligence (AI) to streamline the hiring process. One of the key innovations in this space is the AI Resume Scorecard—an automated system that evaluates and ranks resumes based on predefined criteria such as skills, experience, keywords, and formatting. This technology helps recruiters quickly identify the most suitable candidates while offering job seekers valuable insights into how their resumes align with job requirements. As AI continues to reshape recruitment, understanding how resume scorecards work is essential for both applicants and hiring professionals.

# An AI Resume Scorecard is a tool that uses artificial intelligence to quickly review and rate resumes based on how well they match a job description. It looks at keywords, skills, experience, and formatting to give each resume a score, helping employers find the best candidates faster. This system makes hiring more efficient and fair by applying the same criteria to all resumes, though it may miss some details that a human would notice. It's becoming a popular way to speed up recruitment and help job seekers improve their resumes.

# 

# PROBLEM STATEMENT:

Recruiters often struggle to manually evaluate a large number of resumes, while candidates lack tools to assess how well their resumes match job descriptions. The AI Resume Scorecard project aims to automate resume screening by extracting key information and comparing it with job requirements using AI techniques, providing a match score and improvement suggestions.

**Top of Form**

**Bottom of Form**

**SOLUTION**

* To address the inefficiencies and subjectivity in manual resume screening,
* AI Resume Scorecard proposes an intelligent, automated system that leverages Natural Language Processing (NLP) and Machine Learning (ML) to evaluate resumes against job descriptions.

The system will perform the following:

* Resume Parsing: Extract relevant information such as skills, experience, education, certifications, and project work from resumes in PDF, DOCX, or image formats using OCR and NLP techniques.
* Job Description Analysis: Process the job description to identify key requirements such as mandatory skills, desired qualifications, and relevant experience levels.
* Match Scoring: Use techniques like TF-IDF vectorization and cosine similarity to compare the candidate's resume content with the job description. A match score (out of 100) will be generated to reflect how well the resume aligns with the job.
* Detailed Feedback: Along with the score, the system will provide section-wise insights—highlighting strong areas and suggesting improvements such as missing keywords, better formatting, or more specific achievements.
* User Interface: A user-friendly interface will allow job seekers to upload their resumes and job descriptions and receive a scorecard with visual feedback.

## PROGRAM:

**Index.html:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>AI Resume Scorecard</title>

<link href="https://fonts.googleapis.com/css2?family=Quicksand&family=Roboto:wght@400;700&display=swap" rel="stylesheet" />

<style>

\* {

box-sizing: border-box;

}

body {

font-family: 'Quicksand', sans-serif;

background: #0a0f3d; /\* dark royal blue \*/

display: flex;

justify-content: center;

align-items: center;

min-height: 100vh;

margin: 0;

color: #eee;

overflow-x: hidden;

}

.container {

background: linear-gradient(135deg, #2a1a72, #4c3c94, #3b2c7f);

padding: 40px 35px 50px;

border-radius: 16px;

box-shadow: 0 8px 32px rgba(70, 47, 147, 0.7);

width: 100%;

max-width: 480px;

position: relative;

overflow: hidden;

color: #eee;

}

/\* Shimmer animation on container \*/

.container::before {

content: "";

position: absolute;

top: -50%;

left: -50%;

width: 200%;

height: 200%;

background: linear-gradient(

120deg,

transparent 30%,

rgba(255,255,255,0.12) 50%,

transparent 70%

);

animation: shimmer 4s linear infinite;

pointer-events: none;

z-index: 0;

}

@keyframes shimmer {

0% { transform: rotate(25deg) translateX(-100%); }

100% { transform: rotate(25deg) translateX(100%); }

}

main.container > \* {

position: relative;

z-index: 1;

}

h1 {

text-align: center;

font-size: 2.3rem;

font-weight: 700;

margin-bottom: 18px;

color: #d7d2f0; /\* soft lavender \*/

letter-spacing: 0.04em;

user-select: none;

text-shadow: 0 0 6px #8c80d9;

}

h2.welcome {

text-align: center;

font-weight: 600;

margin-bottom: 32px;

font-size: 1.2rem;

color: #c1bff0;

font-style: italic;

user-select: none;

text-shadow: 0 0 4px #7b6fd1;

}

form {

display: flex;

flex-direction: column;

}

label {

font-weight: 700;

margin-bottom: 8px;

font-size: 1.05rem;

color: #b0aee3;

user-select: none;

}

input[type="file"],

textarea {

padding: 14px 16px;

border-radius: 10px;

border: 1.8px solid #695eb3;

font-family: 'Roboto', sans-serif;

font-size: 1rem;

outline: none;

transition: border-color 0.3s ease, box-shadow 0.3s ease;

resize: none;

color: #eee;

margin-bottom: 22px;

background-color: #2f2770;

cursor: pointer;

user-select: text;

}

input[type="file"] {

cursor: pointer;

}

input[type="file"]::-webkit-file-upload-button {

background: #8c80d9;

color: #23233d;

border: none;

padding: 10px 18px;

border-radius: 12px;

cursor: pointer;

font-weight: 700;

font-family: 'Roboto', sans-serif;

transition: background-color 0.3s ease;

}

input[type="file"]::-webkit-file-upload-button:hover {

background: #7b6fd1;

color: #e0e0ff;

}

input[type="file"]:focus,

textarea:focus {

border-color: #b8afff;

box-shadow: 0 0 12px #b8afffbb;

background-color: #39306e;

color: #f0f0ff;

}

textarea::placeholder {

color: #bbb8d5;

font-style: italic;

}

/\* Interactive label float effect \*/

.input-group {

position: relative;

margin-bottom: 30px;

}

.input-group label {

position: absolute;

top: 14px;

left: 16px;

background: #2a1a72;

padding: 0 6px;

color: #b2aee1;

font-weight: 400;

font-size: 0.9rem;

pointer-events: none;

transition: 0.25s ease all;

border-radius: 4px;

user-select: none;

}

.input-group input:focus + label,

.input-group textarea:focus + label,

.input-group input:not(:placeholder-shown) + label,

.input-group textarea:not(:placeholder-shown) + label {

top: -10px;

left: 12px;

font-size: 0.75rem;

color: #b8afff;

font-weight: 700;

box-shadow: 0 0 6px #b8afffcc;

}

button {

margin-top: 4px;

padding: 15px 0;

background: #8c80d9;

border: none;

border-radius: 24px;

font-weight: 900;

font-size: 1.15rem;

color: #23233d;

cursor: pointer;

overflow: hidden;

position: relative;

transition: background-color 0.3s ease, box-shadow 0.3s ease;

box-shadow: 0 6px 15px rgba(140, 128, 217, 0.7);

user-select: none;

}

button:hover {

background: #7b6fd1;

box-shadow: 0 8px 24px rgba(123, 111, 209, 0.85);

color: #e0e0ff;

}

button:focus {

outline: none;

box-shadow: 0 0 18px #b8afff, 0 0 26px #b8afffcc;

}

button:active {

transform: scale(0.97);

}

/\* Progress bar inside button \*/

button::before {

content: "";

position: absolute;

left: 0;

top: 0;

height: 100%;

width: 0;

background: rgba(255,255,255,0.15);

transition: width 0.4s ease;

z-index: 1;

border-radius: 24px;

}

button:active::before {

width: 100%;

transition: width 0.1s ease;

}

/\* Result styling \*/

.result {

margin-top: 30px;

padding: 24px 20px;

border-radius: 16px;

background: #2f2260;

color: #b8afff;

font-weight: 700;

font-size: 1.3rem;

text-align: center;

box-shadow: 0 4px 18px rgba(138, 118, 200, 0.5);

user-select: none;

animation: fadeInUp 0.7s ease forwards;

position: relative;

user-select: none;

}

.result .score {

font-size: 3rem;

color: #a18aff;

margin-bottom: 8px;

text-shadow: 0 0 10px #a18affcc;

}

.result .message {

font-weight: 600;

font-size: 1.1rem;

color: #c4bfff;

}

@keyframes fadeInUp {

from {

opacity: 0;

transform: translateY(25px);

}

to {

opacity: 1;

transform: translateY(0);

}

}

/\* Responsive \*/

@media (max-width: 520px) {

.container {

padding: 30px 25px 40px;

max-width: 90vw;

}

h1 {

font-size: 1.9rem;

}

h2.welcome {

font-size: 1rem;

}

button {

font-size: 1.05rem;

padding: 14px 0;

}

.result {

font-size: 1.1rem;

}

.result .score {

font-size: 2.4rem;

}

.result .message {

font-size: 1rem;

}

}

</style>

</head>

<body>

<main class="container" role="main" aria-label="AI Resume Scorecard Form">

<h2 class="welcome">Welcome to your professional AI Resume evaluation</h2>

<h1>AI Resume Scorecard</h1>

<form method="POST" action="/score" enctype="multipart/form-data" aria-describedby="form-instructions" novalidate>

<div class="input-group">

<input type="file" id="resume" name="resume" accept=".pdf,.docx,.doc,.jpg,.jpeg,.png" required aria-required="true" placeholder=" " />

<label for="resume">Upload Resume (PDF, DOCX, JPG)</label>

</div>

<div class="input-group">

<textarea id="jd" name="jd" rows="6" placeholder=" " required aria-required="true"></textarea>

<label for="jd">Paste Job Description</label>

</div>

<button type="submit" aria-label="Submit form to evaluate resume">Evaluate</button>

</form>

{% if score %}

<section class="result" role="region" aria-live="polite" aria-atomic="true">

<div class="score">Score: {{ score }}%</div>

<div class="message">{{ message }}</div>

</section>

{% endif %}

</main>

</body>

</html>

**App.py:**

# app.py

from flask import Flask, render\_template, request

from resume\_utils import \*

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

@app.route('/')

def index():

return render\_template('index.html')

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

def score():

resume\_file = request.files['resume']

job\_description = request.form['jd']

ext = resume\_file.filename.split('.')[-1].lower()

resume\_path = f"temp\_resume.{ext}"

resume\_file.save(resume\_path)

resume\_text = ""

if ext == 'pdf':

resume\_text = extract\_text\_from\_pdf(resume\_path)

if len(resume\_text.strip()) < 50:

pages = convert\_from\_path(resume\_path, 300)

for page in pages:

resume\_text += pytesseract.image\_to\_string(page).lower()

elif ext in ['docx', 'doc']:

resume\_text = extract\_text\_from\_docx(resume\_path)

elif ext in ['jpg', 'jpeg', 'png', 'bmp']:

resume\_text = extract\_text\_from\_image(resume\_path)

else:

return "❌ Unsupported file type."

score = calculate\_score(resume\_text, job\_description)

# Feedback message

if score == 100:

message = "✅ Excellent Match – All job keywords found in resume."

elif score >= 80:

message = "✅ Good Match – Your resume matches well with the job description."

elif score >= 40:

message = "⚠ Medium Match – Some keywords matched. Try to add more relevant skills."

else:

message = "❌ Low Match – Your resume has very little alignment with the job description."

return render\_template('index.html', score=score, message=message)

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

app.run(debug=True)

**Resume\_Utils:**

# resume\_utils.py

import os

import re

import nltk

from docx import Document

from PIL import Image

import pytesseract

import pdfplumber

from pdf2image import convert\_from\_path

nltk.download('punkt')

def extract\_text\_from\_pdf(file\_path):

text = ""

try:

with pdfplumber.open(file\_path) as pdf:

for page in pdf.pages:

page\_text = page.extract\_text()

if page\_text:

text += page\_text + "\n"

except:

pass

return text.lower()

def extract\_text\_from\_docx(file\_path):

doc = Document(file\_path)

full\_text = [para.text for para in doc.paragraphs]

return "\n".join(full\_text).lower()

def extract\_text\_from\_image(file\_path):

try:

img = Image.open(file\_path)

text = pytesseract.image\_to\_string(img)

return text.lower()

except:

return ""

def clean\_text(text):

text = re.sub(r'[^a-zA-Z\s]', '', text)

tokens = nltk.word\_tokenize(text)

return " ".join(tokens)

def calculate\_score(resume\_text, job\_description):

cleaned\_resume = clean\_text(resume\_text)

cleaned\_jd = clean\_text(job\_description)

jd\_words = set(cleaned\_jd.split())

resume\_words = set(cleaned\_resume.split())

matched\_words = jd\_words.intersection(resume\_words)

min\_score = 20

total\_keywords = len(jd\_words)

matched = len(matched\_words)

if total\_keywords == 0:

score = 0

elif matched == 0:

score = min\_score

elif matched == total\_keywords:

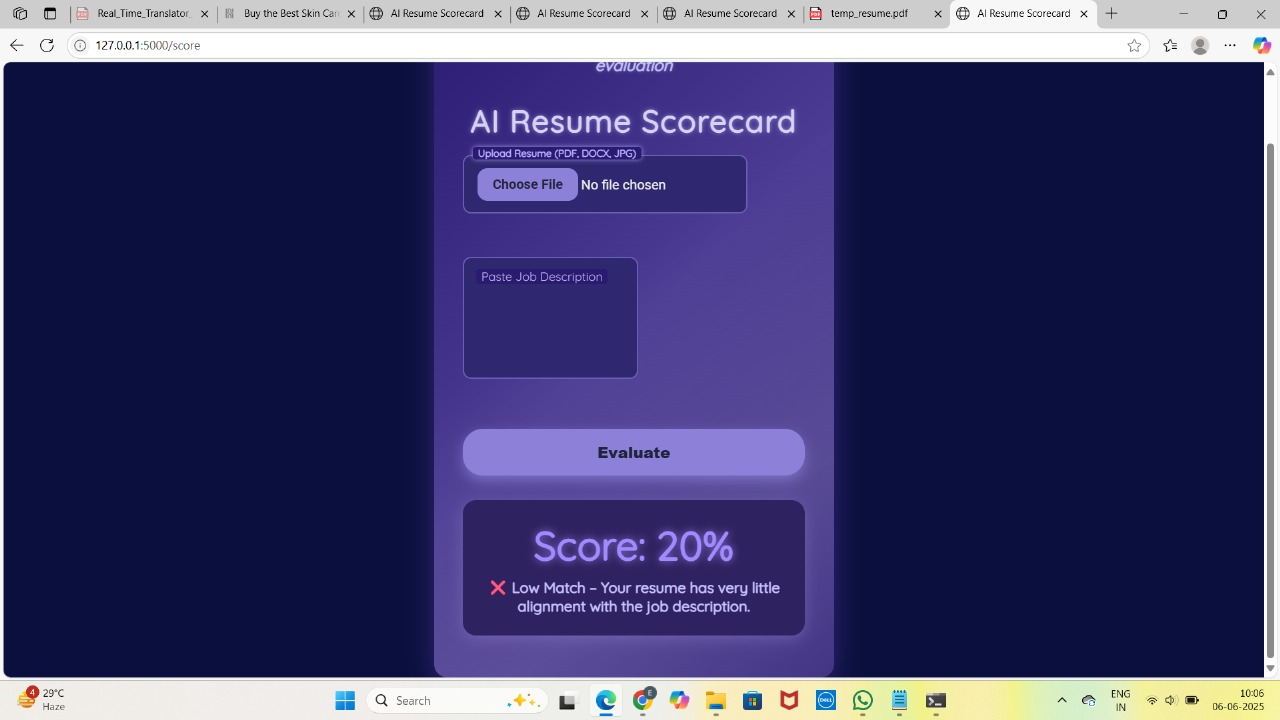
score = 100

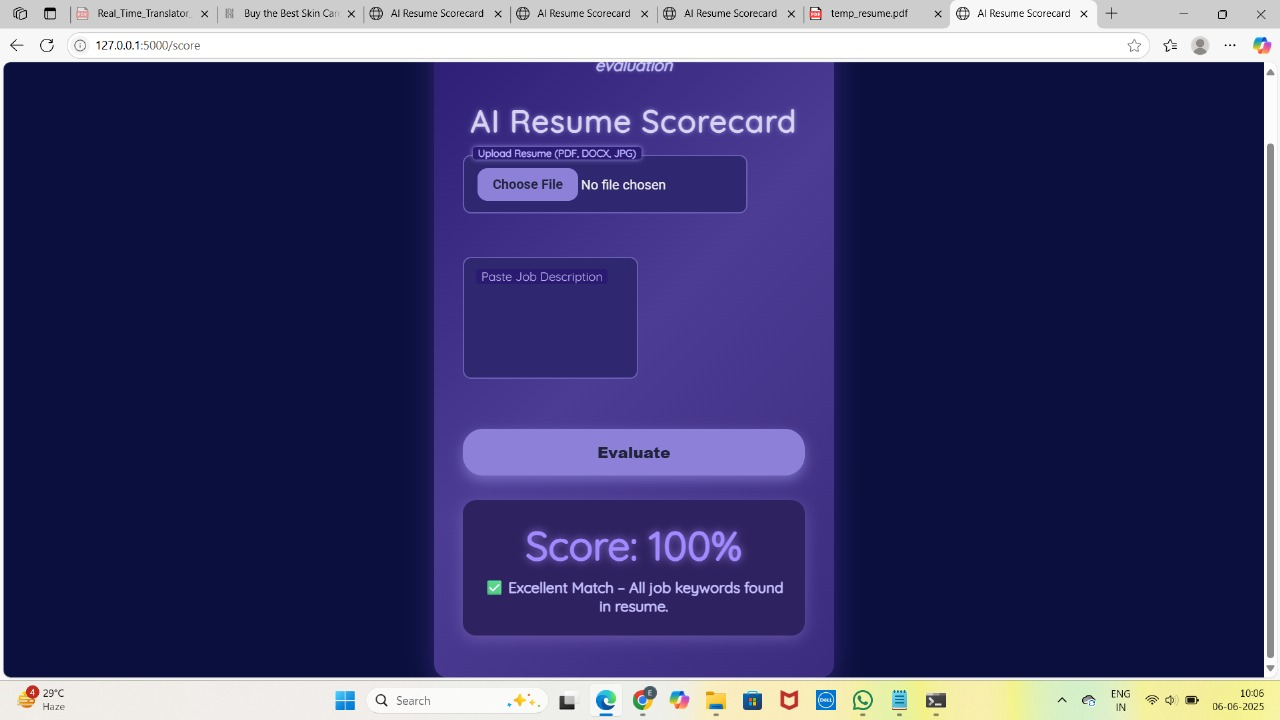
else:

score = min\_score + ((matched / total\_keywords) \* (100 - min\_score))

return round(score)

**OUTPUT:**





**Machine Learning Algorithms**

Machine learning algorithms are essentially sets of instructions that allow computers to learn from data, make predictions, and improve their performance over time without being explicitly programmed.

1. Text Preprocessing

* Tokenize text from both the resume and job description.
* Remove stopwords, punctuations, and perform lowercasing.

1. Feature Extraction using TF-IDF

* TF-IDF (Term Frequency–Inverse Document Frequency) transforms the resume and job description into numerical vectors that represent the importance of terms.

1. Similarity Scoring using Cosine Similarity

* Calculate the cosine similarity between the two TF-IDF vectors to get a match score between 0 and 1.
* Multiply by 100 to convert it into a percentage (e.g., 85% match).