CHAPTER 1

INTRODUCTION

AI-powered recruitment systems are revolutionizing the hiring landscape by integrating advanced technologies to optimize the recruitment process. These systems can analyze vast amounts of data from multiple sources, including social media profiles, job boards, and internal databases, to identify and attract top talent. By leveraging natural language processing (NLP), AI can understand and interpret job descriptions and candidate resumes more effectively, matching candidates to roles with greater precision. One of the notable features of AI in recruitment is its ability to conduct initial candidate assessments.

Video interviewing platforms powered by AI can evaluate non-verbal cues, such as facial expressions and body language, as well as verbal responses to gauge candidate suitability.

This adds an additional layer of analysis that goes beyond traditional resume screening, providing a more comprehensive view of the candidate.AI-powered tools also facilitate diversity and inclusion initiatives by mitigating unconscious biases that can occur in human-led hiring processes. By standardizing evaluations and focusing on objective criteria, these systems help ensure a fairer recruitment process. Moreover, AI can identify patterns and trends that may indicate biases in current hiring practices, allowing organizations to address and correct them proactively. Another advantage of AI-powered recruitment systems is their scalability. They can handle a high volume of applications efficiently, making them ideal for organizations of all sizes, from small startups to large enterprises.

This scalability also allows for consistent and high-quality candidate experiences, regardless of the number of applicants. Furthermore, AI can enhance employee retention by identifying candidates whose skills and personalities align well with the company culture and job requirements. This leads to better job satisfaction and reduces turnover rates.

1.1 Objective

The objective of AI Recruitment System is to empower job seekers by providing them with valuable tools and insights to enhance their job search experience. Specifically, the platform aims to achieve the following objectives:

1. **Salary Insight:** Provide job seekers with transparent salary data and insights across various industries and roles, enabling informed decision-making regarding job opportunities.
2. **Resume Status Notifications:** Notify job seekers promptly about the status of their job applications, ensuring they are aware of progress updates and can take timely actions.
3. **Recruiter Engagement Updates:** Inform job seekers when recruiters download their resumes, offering visibility into recruiter interest and enabling strategic follow-ups.
4. **CV Score Check:** Allow job seekers to check the CV score of their resumes, giving them feedback on their resume quality and suggestions for improvement to increase their chances of being noticed by recruiters.
5. **Enhanced Job Search Efficiency:** Streamline the job application process by reducing uncertainty, improving transparency, and empowering candidates with actionable insights.

1.2 Existing System

Existing recruitment systems primarily depend on manual resume review and simple keyword matching within Applicant Tracking Systems (ATS). This approach is time-consuming, prone to errors, and often overlooks the best candidates. Job seekers must search various platforms to find relevant courses for upskilling, creating inefficiencies. These systems lack integration between resume evaluation, job recommendations, and skill enhancement, leading to a fragmented and less effective recruitment process overall.

The AI-Powered Recruitment System aims to address these challenges by leveraging advanced technologies to provide a more comprehensive, accurate, and user-friendly solution.

1.3 Proposed System

The proposed AI-powered recruitment system automates resume analysis, job matching, and skill recommendations. It utilizes advanced natural language processing to extract key information from resumes and matches candidates with suitable job openings based on their skills and experience. The system also assesses candidates' ATS scores and suggests relevant courses for skill enhancement from a single platform, streamlining the recruitment process for employers and job seekers. This integrated approach improves efficiency, accuracy, and candidate-job fit.

Existing recruitment systems primarily depend on manual resume review and simple keyword matching within Applicant Tracking Systems (ATS). This approach is time-consuming, prone to errors, and often overlooks the best candidates. Job seekers must search various platforms to find relevant courses for upskilling, creating inefficiencies. These systems lack integration between resume evaluation, job recommendations, and skill enhancement, leading to a fragmented and less effective recruitment process overall.

The proposed system leverages advanced machine learning algorithms and natural language processing to create a robust AI-powered recruitment tool. By integrating resume parsing and job matching capabilities, it provides precise skill extraction, detailed job recommendations, and eligibility scoring. The system will include enhanced features such as generating ATS (Applicant Tracking System) scores to evaluate resume compatibility and recommending relevant courses through website links for skill enhancement. A dedicated results page will present comprehensive output, including recommended courses and resume parsing details, improving user experience and actionable insights for job seekers.

CHAPTER 2

SYSTEM REQUIREMENTS SPECIFICATION

It gives the information regarding analysis done for the proposed system. System Analysis is done to capture the requirement of the user of the proposed system. It also provides the information regarding the existing system and the need for the proposed system. The key features of the proposed system and the requirement specifications of the proposed system are discussed below.

2.1 Hardware Requirements

Processor type	:	Intel i3 or more faster processors
Processor speed	:	Minimum 2.4 GHz or faster
RAM	:	8GB or more.
HARD DISK	:	16 GB hard disk recommended.
Monitor	:	VGA or higher resolution 800x600 or higher resolution
Pointing device	:	Microsoft Mouse or compatible pointing device
Keyboard	:	Standard 102 Keys.

2.2 Software Requirements

Operating system	:	Windows 7 or later
Front End	:	HTML,CSS,JAVASCRIPT
Back End	:	PYTHON
Scripting Language	:	FLASK

2.3 Functional Requirements

The system should facilitate users in uploading their resumes in various formats, including PDF, DOC, and DOCX, and parse these documents accurately to extract and analyze relevant information. It must be capable of identifying and extracting key skills from the resumes and calculating an Applicant Tracking System (ATS) score based on keyword matching and text relevance. The system should then match the extracted resume data with predefined job descriptions to recommend suitable job titles and provide detailed descriptions for each recommendation. Additionally, the system should predict potential salaries for the recommended job titles, using a predefined salary range dataset. The user interface should be designed to present all results clearly on a dedicated results page. The system should offer actionable insights, enabling users to improve their resumes and better match their skills with job opportunities. Furthermore, it should handle errors gracefully and provide informative feedback to users.

2.4 Non - Functional Requirements

The non-functional requirements of an AI recruitment system include scalability to handle varying workloads and user demands efficiently, ensuring high availability and reliability for uninterrupted service. Security is paramount, encompassing data encryption, access control, and compliance with data protection regulations. The system must offer a user-friendly interface with fast response times for an optimal user experience. Interoperability with existing HR systems is essential for seamless integration. Additionally, it should include robust error handling and logging mechanisms to facilitate maintenance and troubleshooting, while maintaining accuracy and fairness in decision-making to uphold ethical standards.

CHAPTER 3

SYSTEM DESIGN

The process of design involves "conceiving and planning out in mind and making a drawing, pattern or a sketch". The system design transforms a logical representation of what a given system is required to do into the physical reality during development. Important design factors such as reliability, response time, throughput of the system, maintainability, expandability etc., should be taken into account. Design constraints like cost, hardware limitations, standard compliance etc., should also be dealt with it.

3.1 Data Flow Diagram

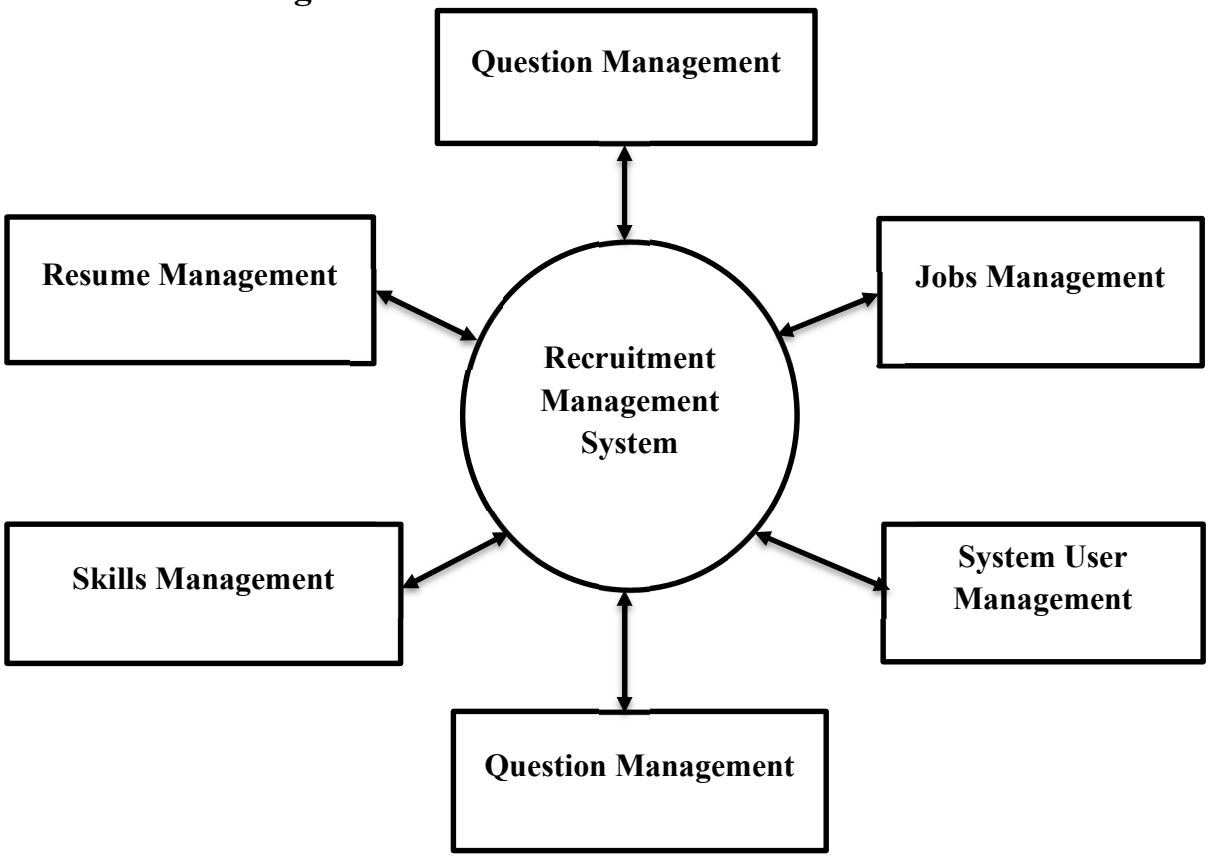

Figure 3.1 DFD for AI Recruitment System

Figure 3.1 depicts a job portal system's process, including login, registration, search, and user management.

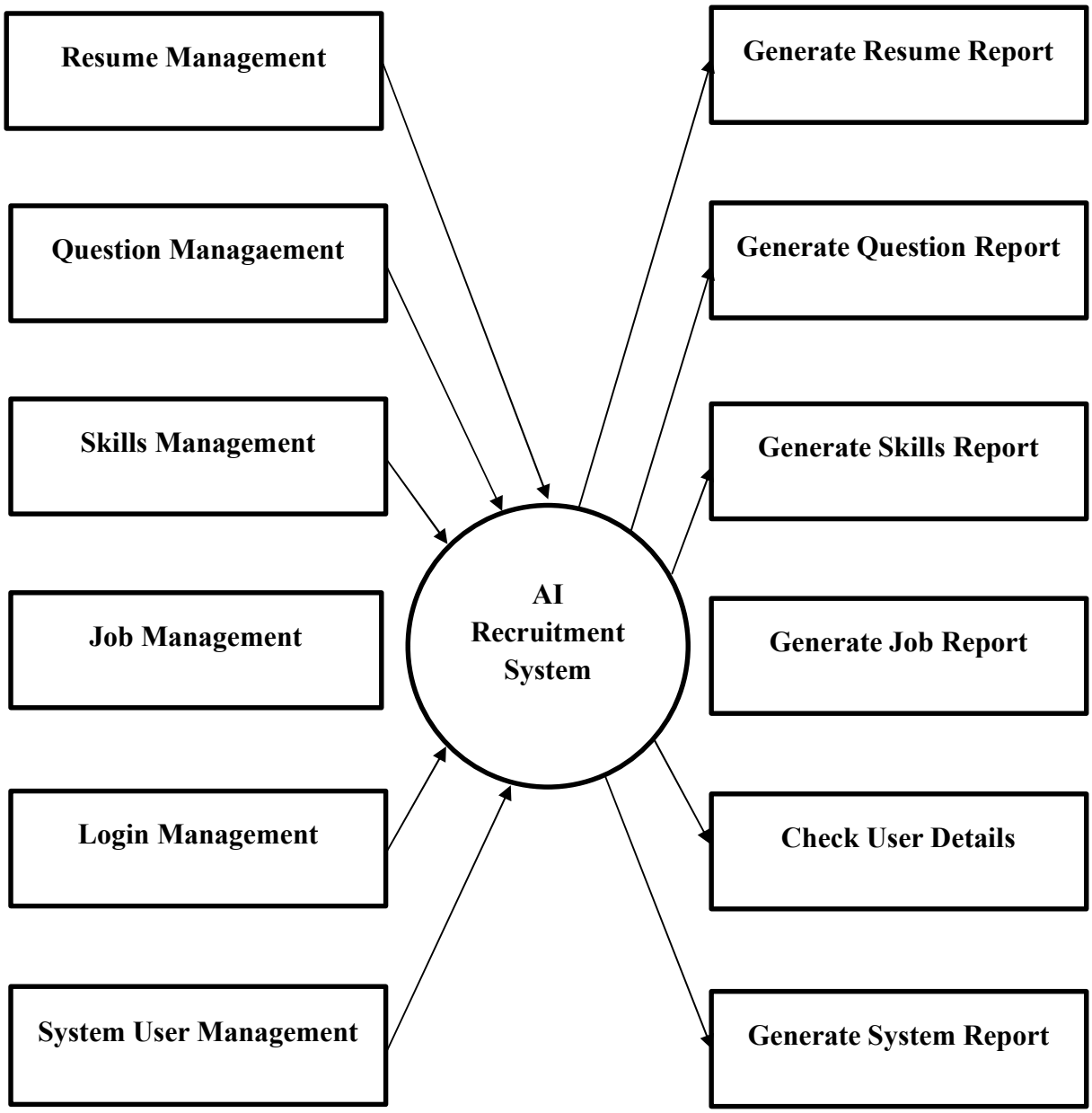

Figure 3.2 Second Level DFD for AI Recruitment System

Figure 3.2 illustrates detailed processes: user login verification, job seeker registration, resume upload, employer registration, job posting, and searching for job vacancies.

3.2 System Architecture

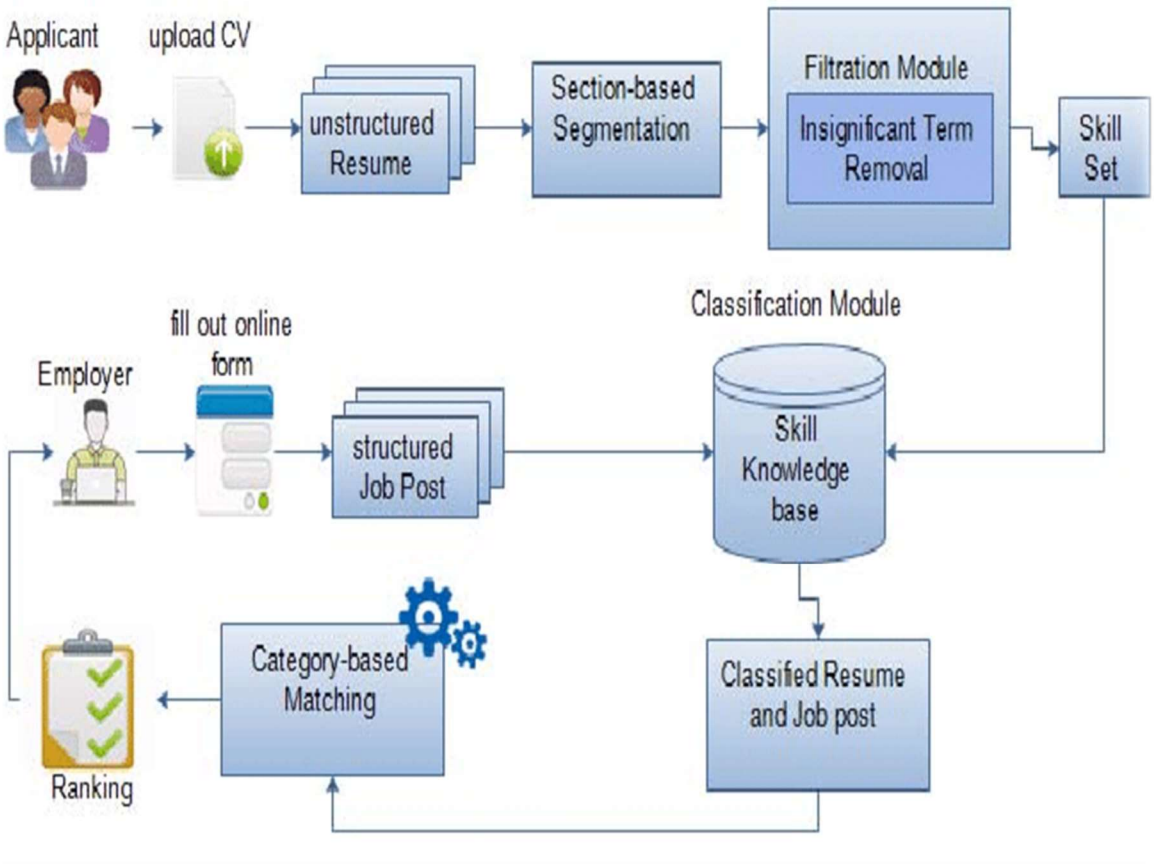

Fig 3.3 System Architecture of AI Recruitment System

The AI-Powered Recruitment System features a Flask backend and a user-friendly frontend built with HTML, CSS, and JavaScript. Users upload resumes, which are processed using PyPDF2 and docx2txt to extract text. The text is then preprocessed with NLP techniques and converted into numerical vectors using TF-IDF. These vectors, along with additional features like skills and experience, are input into a Random Forest classifier to predict the most suitable job category. The prediction results are displayed to users via the web interface, ensuring efficient and accurate resume categorization.

CHAPTER 4
SYSTEM IMPLEMENTATION

4.1 Module Implementation

For a Job Connect website, four major modules can be implemented to ensure comprehensive functionality and user management:

1. **The Data Collection and Parsing Module:** Serves as the foundation, gathering and analyzing job postings to extract specific skills, qualifications, and requirements. Simultaneously, it parses candidate resumes and profiles to extract essential information such as skills, experience, and education, facilitating automated matching with job descriptions.
2. **Job Posting and Management Module:** Designed for recruiters, this module facilitates the posting of job vacancies and the management of existing job listings. It includes forms for creating new job entries, a dashboard for overseeing and managing job postings, tools for tracking and reviewing applications, and options for editing or deleting job listings.
3. **The Pre-screening and Filtering Module:** Employs advanced natural language processing (NLP) techniques to automatically screen resumes and match candidate qualifications with job requirements. This module effectively filters out candidates who do not meet minimum criteria, ensuring that only qualified applicants proceed to subsequent stages.
4. **Candidate Ranking and Scoring Module**: Utilizes machine learning algorithms to objectively rank and score candidates based on their alignment with job criteria. This AI-driven approach enhances efficiency and predicts candidate success and performance based on historical data and benchmarks. Each module will have specific views, controllers, and models tailored to the framework and technologies used in website development.

4.2 TOOL IMPLEMENTATIONS

This chapter of the report describes the Functions, packages, and modules used in the project:

4.2.1 LIBRARIES AND FRAMEWORKS PYTHON FLASK

Python is a high-level programming language known for its readability and versatility. Flask is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It handles much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel.

1. **HTML**

   Hypertext Markup Language is the standard markup language for documents designed to be displayed in a web browser. It provides the structure for web pages using a system of tags and attributes.

2. **CSS**

   Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language like HTML. It controls the layout and appearance of multiple web pages all at once.

3. **JavaScript**

   JavaScript is a programming language that enables interactive web pages and is an essential part of web development. It allows for dynamic content, animation, and user interaction.

4. **Flask**

   Flask is a high-level Python web framework that enables rapid development of secure and maintainable websites. It provides tools and libraries for common web development tasks, such as URL routing, template rendering, and database access.

4.2.2 FUNCTIONAL MODULES

The functional modules included in the project are listed below:

1. **UPLOAD MODULE**

   This module allows users to upload resumes in various formats such as CSV, PDF, and Excel. It ensures the uploaded files meet specified format and size requirements, providing a seamless way to input resume data into the system.

2. **PARSING MODULE**

   The parsing module is responsible for extracting text and key information from the uploaded resumes. It handles different file formats efficiently and employs techniques to parse and organize the data, including names, contact details, education, and work experience.

3. **DATA EXTRACTION MODULE**

   Utilizing Natural Language Processing (NLP) techniques, this module extracts and structures relevant information from resumes. It identifies important entities such as skills, job titles, companies, and dates, ensuring that all critical information is captured accurately.

4. **CLASSIFICATION MODULE**

   This module classifies resumes based on predefined categories, such as job roles and skill sets, using AI and machine learning models. It continuously trains on labeled data to improve the accuracy of the classification, helping to streamline the candidate screening process.

5. **SCORING MODULE**

   This module assigns scores to resumes based on their relevance to specific job postings or criteria defined by recruiters. It ranks resumes, making it easier for recruiters to identify the most suitable candidates for a position.

6. **EXPORT MODULE**

   The export module allows users to export the extracted and classified data to various formats like CSV and Excel. It also supports generating custom reports based on user-defined templates, facilitating easy sharing and analysis of recruitment data.

6.3 SOURCE CODE

6.3.1 HOMEPAGE

```
<!DOCTYPE html>
<html lang="en">
<head>
          <meta charset="UTF-8">
          <meta name="viewport" content="width=device-width, initial-scale=1.0">
          <title>NAS Hunters</title>
          <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
          <style>
              /* General body styling */
      body {
          overflow: hidden;
          background: #f0f0f0; /* Background color or image */
          position: relative;
      }

      /* Bubble container */
      /* More Specific */
      .container .bubbles div {
          position: absolute;
          top: 0;
          left: 0;
          width: 100%;
          height: 100%;
          pointer-events: none;
          overflow: hidden;
      }

      /* Individual bubble styling */
      .bubbles div {
          position: absolute;
          border-radius: 50%;
          background: rgba(255, 255, 255, 0.7);
          opacity: 0.5;
          animation: bubbleAnimation 10s infinite;
      }

      /* Bubble animation */
      @keyframes bubbleAnimation {
          0% {
              transform: translateY(0) scale(0.5);
```

```
            opacity: 0.5;
        }
        50% {
            transform: translateY(-100vh) scale(1);
            opacity: 0.8;
        }
        100% {
            transform: translateY(0) scale(0.5);
            opacity: 0.5;
        }
    }

    /* Specific bubbles */
    .bubbles div:nth-child(1) {
        width: 40px;
        height: 40px;
        left: 10%;
        animation-duration: 12s;
    }

    .bubbles div:nth-child(2) {
        width: 60px;
        height: 60px;
        left: 30%;
        animation-duration: 10s;
    }

    .bubbles div:nth-child(3) {
        width: 30px;
        height: 30px;
        left: 50%;
        animation-duration: 8s;
    }

    /* Add more specific bubbles if needed */

            header {
                text-align: center;
                padding: 20px;
                background-color: #3e9d46;
                color: white;
                font-size: 2rem;
                animation: headerAnimation 5s ease-in-out infinite;
  /* Slow-motion animation */
                margin-bottom: 20px; /* Space below header */
            }
```

```
92.
93.             @keyframes headerAnimation {
94.                 0% {
95.                     transform: scale(1);
96.                     opacity: 1;
97.                 }
98.                 50% {
99.                     transform: scale(1.1);
100.                    opacity: 0.8;
101.                }
102.                100% {
103.                    transform: scale(1);
104.                    opacity: 1;
105.                }
106.            }
107.
108.            .container {
109.                max-width: 800px;
110.                margin: 20px auto;
111.                padding: 20px;
112.                background-color: white;
113.                border-radius: 8px;
114.                box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
115.                text-align: center;
116.            }
117.
118.            .container h2 {
119.                margin-top: 0;
120.            }
121.
122.            .btn {
123.                display: inline-block;
124.                padding: 10px 20px;
125.                background-color: #007bff;
126.                color: white;
127.                text-decoration: none;
128.                border-radius: 5px;
129.                font-size: 16px;
130.                transition: background-color 0.3s, transform 0.2s;
131.            }
132.
133.            .btn:hover {
134.                background-color: #0056b3;
135.                transform: scale(1.05);
136.            }
137.
```

```
            .services-section {
                margin-top: 20px;
            }

            .services-section h2 {
                font-size: 1.5rem;
                margin-bottom: 10px;
            }

            .services {
                display: flex;
                flex-wrap: wrap;
                justify-content: center;
                gap: 20px;
            }

            .service {
                background-color: #f9f9f9;
                border: 1px solid #ddd;
                border-radius: 8px;
                padding: 15px;
                width: calc(50% - 20px);
                box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
                transition: box-shadow 0.3s, transform 0.3s;
            }

            .service:hover {
                box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
                transform: translateY(-5px);
            }

            .service h3 {
                margin-top: 0;
            }
        </style>
    </head>
    <body>
        <header>
            NAS Hunters
        </header>
        <div class="container">
            <h2>Welcome to NAS Hunters Recruitment System</h2>
            <p>Your AI-powered recruitment assistant.</p>
            <a href="/start" class="btn">START</a>

            <div class="services-section">
```

```
                <h2>Our Key Services</h2>
                <div class="services">
                    <div class="service">
                        <h3>Resume Analysis</h3>
                        <p>Analyze resumes using advanced AI
   algorithms.</p>
                    </div>
                    <div class="service">
                        <h3>Job Prediction</h3>
                        <p>Predict suitable job categories for
   candidates.</p>
                    </div>
                    <div class="service">
                        <h3>ATS Scoring</h3>
                        <p>Evaluate resumes with ATS compatibility
   scoring.</p>
                    </div>
                </div>
            </div>
        </div>
    </body>
    </html>
```

202.3.1 RESUME ANALYSIS

```
    <!DOCTYPE html>
    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
   scale=1.0">
        <title>Resume Analysis</title>
        <link rel="stylesheet" href="{{ url_for('static',
   filename='style.css') }}">
    </head>
    <body>
        <div class="container">
            <div class="card">
                <div class="card-header">
                    Resume Analysis
                </div>
                <div class="card-body">
                    <nav>
                        <ul>
                            <li><a href="{{ url_for('home')
   }}">Home</a></li>
                            <li><a href="{{ url_for('about')
   }}">About</a></li>
```

```
                            <li><a href="{{ url_for('contact') }}">Contact</a></li>
                        </ul>
                    </nav>
                    <form id="resume-form" action="{{ url_for('predict') }}" method="post" enctype="multipart/form-data">
                        <div class="form-group">
                            <label for="resume">Upload Resume:</label>
                            <input type="file" id="resume" name="resume_file" accept=".pdf,.doc,.docx" required>
                        </div>
                        <div class="form-group">
                            <label for="skills">Skills (comma separated):</label>
                            <input type="text" id="skills" name="skills">
                        </div>
                        <div class="form-group">
                            <button type="submit">Analyze</button>
                        </div>
                    </form>
                    <div id="loading" style="display: none;">Loading...</div>
                    <div id="error" style="display: none;"></div>
                </div>
            </div>
        </div>
    </body>
    </html>
```

4.3.3 CONTACT US PAGE

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Contact - NAS HUNTERS</title>
    <style>
        body {
```

```
        font-family: Arial, sans-serif;
        background: linear-gradient(to right, #6a11cb, #2575fc);
        color: #fff;
        padding: 20px;
        text-align: center;
    }
    h1 {
        color: #fff;
        animation: fadeInDown 1s ease-in-out;
    }
    form {
        background-color: rgba(255, 255, 255, 0.2);
        padding: 20px;
        border-radius: 10px;
        border: 2px solid #fff;
        width: 50%;
        margin: 0 auto;
        animation: fadeInUp 1.5s ease-in-out;
    }
    label, input, textarea {
        display: block;
        width: 100%;
        margin: 10px 0;
    }
    input, textarea {
        padding: 10px;
        border-radius: 5px;
        border: 1px solid #fff;
    }
    button {
```

```
    padding: 10px 20px;
    border: none;
    background-color: #4CAF50;
    color: #fff;
    border-radius: 5px;
    cursor: pointer;
}
button:hover {
    background-color: #45a049;
}
.contact-details {
    margin-top: 30px;
    padding: 20px;
    border: 2px solid #fff;
    border-radius: 10px;
    background-color: rgba(255, 255, 255, 0.2);
}
.contact-details h2 {
    margin-top: 0;
    color: #fff;
    animation: fadeInDown 1s ease-in-out;
}
@keyframes fadeInDown {
    from { transform: translateY(-20px); opacity: 0; }
    to { transform: translateY(0); opacity: 1; }
}
@keyframes fadeInUp {
    from { transform: translateY(20px); opacity: 0; }
    to { transform: translateY(0); opacity: 1; }
}
```

```
    </style>
    <script>
        async function sendMessage(event) {
            event.preventDefault();
            const name = document.querySelector('input[name="name"]').value;
            const message =
document.querySelector('textarea[name="message"]').value;
            try {
                const response = await fetch('/contact', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/json',
                    },
                    body: JSON.stringify({ name, message }),
                });
                const result = await response.json();
                if (result.success) {
                    alert('Message sent successfully!');
                } else {
                    alert('Failed to send message. Error: ' + result.error);
                }
            } catch (error) {
                console.error('Error:', error);
                alert('Failed to send message.');
            }
        }
    </script>
</head>
<body>
    <h1>Contact Us</h1>
```

```
    <form onsubmit="sendMessage(event)">
        <label for="name">Name:</label>
        <input type="text" name="name" required
        <label for="message">Message:</label>
        <textarea name="message" rows="4" required></textarea>
        <button type="submit">Send Message</button>
    </form>
    <div class="contact-details">
        <h2>Contact Information</h2>
        <p><strong>Ayush Shekar</strong><br>Phone: 9113217988</p>
        <p><strong>K Shreyank</strong><br>Phone: 9597653466</p>
        <p><strong>NagaSimha N</strong><br>Phone: 9901678694</p>
    </div>
</body>
</html>
```

4.3.4 APP.PY

```
from flask import Flask, request, jsonify, render_template,flash, redirect,
url_for, session, send_file
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics.pairwise import cosine_similarity
import nltk
from nltk.corpus import stopwords
import PyPDF2
import docx2txt
import logging
import re
import nexmo
```

```
from fpdf import FPDF
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
app = Flask(__name__)
app.secret_key = 'your_secret_key_here'
# Setup logging
logging.basicConfig(level=logging.INFO)
client = nexmo.Client(key='c8bb823f', secret='FOmLiWYHbfk4adFa')
creators = {
   'Ayush Shekar': '+9113217988',
   'K Shreyank': '+919597653466',
   'NagaSimha N': '+919901678694'
}
# Load and preprocess the data
df = pd.read_csv('Resume.csv')
df.fillna('', inplace=True)
def preprocess_text(text):
   text = text.lower()
   text = re.sub(r'[^\w\s]', '', text)
   words = text.split()
   words = [word for word in words if word not in stop_words]
   return ' '.join(words)
df['Resume_str'] = df['Resume_str'].apply(preprocess_text)
vectorizer = TfidfVectorizer(max_features=1000)
X_text = vectorizer.fit_transform(df['Resume_str']).toarray()
X_skills = df['Category'].apply(lambda x: len(x.split(','))).values.reshape(-
1, 1)
# Dummy columns for 'experience' and 'education'
X_experience = np.zeros((df.shape[0], 1))  # Dummy column for
'experience'
```

```
X_education = np.zeros((df.shape[0], 1))  # Dummy column for
'education'
X = np.hstack((X_text, X_skills, X_experience, X_education))
y = df['Category']  # Assuming 'Category' is what we want to predict
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X, y)
def calculate_ats_score(text):
   keywords = ["python", "machine learning", "data analysis", "flask",
"nlp", "wordpress", "full stack developer", "css", "data
structures","sql"]
   score = 0
   word_count = len(text.split())
   for keyword in keywords:
      if keyword in text:
         score += 10
   if 500 <= word_count <= 1000:
      score += 20
   elif word_count > 1000:
      score += 10
   score = min(score, 100)
   return score
job_descriptions = pd.DataFrame({
   'Job Title': ['Data Scientist', 'Software Engineer', 'Product Manager',
'Sales Manager', 'Digital Marketing Specialist', 'Network Administrator',
'Human Resources Manager', 'Financial Analyst', 'UX/UI Designer',
'Business Analyst', 'Machine Learning Engineer', 'AI Engineer'],
   'Description': [
      '''A Data Scientist collects, analyzes, and interprets large datasets to
identify patterns, trends, and insights. They build and validate predictive
models, perform statistical analysis, and work with machine learning
```

algorithms to improve data-driven decision-making. Responsibilities include data cleaning, data visualization, and communicating findings to stakeholders. Proficiency in programming languages like Python or R, and experience with data visualization tools like Tableau or Power BI are essential.'",

'"A Software Engineer designs, develops, tests, and maintains software applications. They write clean, efficient, and maintainable code in various programming languages such as Java, C++, or Python. They collaborate with cross-functional teams to define requirements, perform code reviews, and troubleshoot/debug issues. Knowledge of software development methodologies, version control systems like Git, and familiarity with frameworks and libraries relevant to their tech stack is crucial.'",

'"A Product Manager is responsible for the strategy, roadmap, and feature definition of a product. They gather and prioritize product and customer requirements, define the product vision, and work closely with engineering, sales, marketing, and support teams to ensure revenue and customer satisfaction goals are met. Strong analytical skills, an understanding of market trends, and the ability to communicate effectively across teams are key attributes.'",

'"A Sales Manager leads and motivates a team of sales representatives to meet and exceed sales targets. They develop and implement strategic sales plans, analyze sales data, and forecast future sales. Responsibilities include training and coaching the sales team, building and maintaining customer relationships, and ensuring customer satisfaction. Strong leadership, excellent communication skills, and the ability to analyze market trends and competitor strategies are essential.'",

'"A Digital Marketing Specialist plans, executes, and manages digital marketing campaigns across various online platforms. They optimize content for SEO, manage PPC campaigns, utilize social media marketing,

and analyze performance metrics to drive growth. Knowledge of tools like Google Analytics, AdWords, and social media management platforms is crucial. Creativity, analytical skills, and staying updated with digital marketing trends are key attributes.'",

'"A Network Administrator is responsible for maintaining the company's IT network, servers, and security systems. They install, configure, and troubleshoot network hardware and software, ensure network security, and optimize network performance. Proficiency in network management tools, knowledge of networking protocols, and experience with firewalls, routers, and switches are important. Strong problem-solving skills and the ability to handle network emergencies are essential.'",

'"A Human Resources Manager oversees the recruitment, training, and development of employees. They manage employee relations, ensure compliance with labor laws, and develop HR policies and procedures. Responsibilities include performance management, employee engagement, and handling disciplinary actions. Strong interpersonal skills, knowledge of HR software, and the ability to handle sensitive situations are crucial.'",

'"A Financial Analyst analyzes financial data to assist in decision-making processes. They create financial models, forecast future performance, and evaluate investment opportunities. Responsibilities include preparing financial reports, analyzing market trends, and providing insights to management. Proficiency in Excel, knowledge of financial software, and strong analytical skills are essential.'",

'"A UX/UI Designer creates user-friendly and visually appealing interfaces for websites and applications. They conduct user research, develop wireframes and prototypes, and collaborate with developers to implement designs. Knowledge of design tools like Sketch, Adobe XD, and Figma, and an understanding of user-centered design principles are

```
crucial. Creativity, attention to detail, and the ability to understand user needs are key attributes.'",
    '''A Business Analyst works with stakeholders to identify business needs and develop solutions. They gather and document requirements, analyze processes, and ensure that IT and business teams are aligned. Responsibilities include creating detailed business analysis, outlining problems, opportunities, and solutions, and supporting project implementation. Strong analytical skills, knowledge of business process modeling, and effective communication skills are essential.''',
    '''A Machine Learning Engineer designs and implements machine learning models and algorithms to solve complex problems. They work on feature extraction, model training, and evaluation, and integrate models into production systems. Proficiency in programming languages such as Python or Java, experience with machine learning frameworks like TensorFlow or PyTorch, and knowledge of data preprocessing and model tuning are crucial.''',
    '''An AI Engineer develops and deploys artificial intelligence models and systems. They work on designing AI algorithms, training models, and applying machine learning techniques to real-world problems. Experience with AI tools and frameworks, understanding of neural networks and deep learning, and proficiency in programming languages such as Python or Java are essential. AI Engineers also collaborate with cross-functional teams to integrate AI solutions into business processes.'''
  ]
})

known_skills = set([
  "python", "machine learning", "data analysis", "flask", "nlp", "wordpress",
```

```
    "full stack developer", "css", "javascript", "sql", "html", "java",
"c++",
    "r", "tableau", "power bi", "excel", "apache", "spark", "aws",
"azure", "docker",
    "data structures",
])

def extract_skills(text):
    text = preprocess_text(text)
    words = set(text.split())
    skills = known_skills.intersection(words)
    return list(skills)

def suggest_job(resume_text, job_descriptions):
    vectorizer = TfidfVectorizer().fit_transform([resume_text] +
job_descriptions['Description'].tolist())
    cosine_sim = cosine_similarity(vectorizer[0:1], vectorizer[1:]).flatten()
    best_job_index = cosine_sim.argmax()
    return job_descriptions['Job Title'][best_job_index]

def predict_salary(job_title):
    salary_dict = {
        'Data Scientist': '100,000 - 150,000 USD',
        'Software Engineer': '80,000 - 120,000 USD',
        'Product Manager': '90,000 - 130,000 USD',
        'Sales Manager': '70,000 - 110,000 USD',
        'Digital Marketing Specialist': '60,000 - 100,000 USD',
        'Network Administrator': '70,000 - 90,000 USD',
        'Human Resources Manager': '70,000 - 110,000 USD',
        'Financial Analyst': '60,000 - 90,000 USD',
```

```
        'UX/UI Designer': '70,000 - 110,000 USD',
        'Business Analyst': '60,000 - 90,000 USD',
        'Machine Learning Engineer': '100,000 - 150,000 USD',
        'AI Engineer': '100,000 - 150,000 USD'
    }
    return salary_dict.get(job_title, 'N/A')
def extract_contact_info(resume_text):
    email = re.findall(r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-
z]{2,}\b', resume_text)
    phone = re.findall(r'\b(?:\+?(\d{1,3}))?[-.\s]?(\d{3})[-.\s]?(\d{3})[-
.\s]?(\d{4})\b', resume_text)
    # Format phone numbers correctly
    formatted_phone_numbers = [''.join(number) for number in phone]
    return {'email': email, 'phone': formatted_phone_numbers}
def recommend_courses(missing_skills):
    # This function returns a dictionary of course websites
    course_websites = {
        'Coursera': 'https://www.coursera.org',
        'edX': 'https://www.edx.org',
        'Udemy': 'https://www.udemy.com',
        'LinkedIn Learning': 'https://www.linkedin.com/learning',
        'Pluralsight': 'https://www.pluralsight.com'
    }
    return course_websites
@app.route('/')
def home():
    return render_template('index.html')  # This will now render the
previous 'start.html'
@app.route('/start')
def start():
```

```
    return render_template('home.html')
@app.route('/predict', methods=['POST'])
def predict():
    try:
        file = request.files.get('resume_file')
        if not file:
            return jsonify({'error': 'No file uploaded'}), 400
        resume_text = ''
        if file.filename.endswith('.pdf'):
            try:
                pdf_reader = PyPDF2.PdfReader(file)
                for page_num in range(len(pdf_reader.pages)):
                    page = pdf_reader.pages[page_num]
                    resume_text += page.extract_text()
            except Exception as e:
                app.logger.error(f"Error reading PDF file: {e}")
                return jsonify({'error': 'Error reading PDF file'}), 500
        elif file.filename.endswith(('.doc', '.docx')):
            try:
                resume_text = docx2txt.process(file)
            except Exception as e:
                app.logger.error(f"Error reading DOC/DOCX file: {e}")
                return jsonify({'error': 'Error reading DOC/DOCX file'}), 500
        else:
            return jsonify({'error': 'Unsupported file format'}), 400
        resume_text = preprocess_text(resume_text)
        resume_vector = vectorizer.transform([resume_text]).toarray()
        skills_vector = np.array([len(resume_text.split(','))]).reshape(-1, 1)
        experience_vector = np.zeros((1, 1))  # Dummy value for 'experience'
        education_vector = np.zeros((1, 1))  # Dummy value for 'education'
```

```
        input_vector = np.hstack((resume_vector, skills_vector,
    experience_vector, education_vector))
        prediction = model.predict(input_vector)[0]
        ats_score = calculate_ats_score(resume_text)
        extracted_skills = extract_skills(resume_text)
        suggested_job = suggest_job(resume_text, job_descriptions)
        job_description = job_descriptions.loc[job_descriptions['Job Title']
    == suggested_job, 'Description'].values[0]
        eligibility = ats_score >= 30
        salary_prediction = predict_salary(suggested_job)
        contact_info = extract_contact_info(resume_text)
        return redirect(url_for('results',
                                prediction=prediction,
                                ats_score=ats_score,
                                skills=','.join(extracted_skills),
                                suggested_job=suggested_job,
                                job_description=job_description,
                                eligibility=eligibility,
                                salary_prediction=salary_prediction,
                                email=','.join(contact_info['email']),
                                phone=','.join(contact_info['phone'])))
    except Exception as e:
        app.logger.error(f"Prediction error: {e}")
        return jsonify({'error': 'Prediction failed'}), 500
@app.route('/about')
def about():
    return render_template('about.html')
@app.route('/contact', methods=['GET', 'POST'])
def contact():
    if request.method == 'POST':
```

```
            try:
                data = request.get_json()
                name = data.get('name')
                message = data.get('message')
                full_message = f"From: {name}\nMessage: {message}"
                # Send SMS to each creator
                for creator, phone in creators.items():
                    logging.info(f"Sending message to {creator} (Phone: {phone})")
                    response = client.send_message({
                        'from': 'Nexmo',
                        'to': phone,
                        'text': full_message,
                    })
                    logging.info(f"Message response: {response}")
                return jsonify({'success': True})
            except Exception as e:
                logging.error(f"Error sending message: {e}")
                return jsonify({'success': False, 'error': str(e)})
        return render_template('contact.html')
    @app.route('/results')
    def results():
        prediction = request.args.get('prediction')
        ats_score = request.args.get('ats_score')
        skills = request.args.get('skills')
        suggested_job = request.args.get('suggested_job')
        job_description = request.args.get('job_description')
        eligibility = request.args.get('eligibility')
        salary_prediction = request.args.get('salary_prediction')
        email = request.args.get('email')
        phone = request.args.get('phone')
```

```
        return render_template('results.html',
                               prediction=prediction,
                               ats_score=ats_score,
                               skills=skills,
                               suggested_job=suggested_job,
                               job_description=job_description,
                               eligibility=eligibility,
                               salary_prediction=salary_prediction,
                               email=email,
                               phone=phone)
    @app.route('/download_pdf')
    def download_pdf():
        results = session.get('results', {})
        if not results:
            return jsonify({'error': 'No results to download'}), 400
        try:
            pdf = FPDF()
            pdf.add_page()
            pdf.set_font("Arial", size=12)
            pdf.cell(200, 10, txt="Resume Analysis Report", ln=True,
    align="C")
            pdf.ln(10)
            pdf.cell(200, 10, txt=f"Predicted Job Title: {results['prediction']}",
    ln=True)
            pdf.cell(200, 10, txt=f"ATS Score: {results['ats_score']}", ln=True)
            pdf.cell(200, 10, txt=f"Extracted Skills: {', '.join(results['skills'])}",
    ln=True)
            pdf.cell(200, 10, txt=f"Predicted Salary:
    {results['predicted_salary']}", ln=True)
            pdf.ln(10)
```

```
            pdf.multi_cell(200, 10, txt=f"Job Description:
        {results['job_description']}", align="L")
            pdf_file = '/mnt/data/report.pdf'
            pdf.output(pdf_file)
            return send_file(pdf_file, as_attachment=True)
        except Exception as e:
            logging.error(f"Error generating PDF: {str(e)}")
            return jsonify({'error': 'Failed to generate PDF'}), 500
    @app.route('/download_csv')
    def download_csv():
        results = session.get('results', {})
        if not results:
            return jsonify({'error': 'No results to download'}), 400
        try:
            csv_file = '/mnt/data/report.csv'
            df = pd.DataFrame([results])
            df.to_csv(csv_file, index=False)
            return send_file(csv_file, as_attachment=True)
        except Exception as e:
            logging.error(f"Error generating CSV: {str(e)}")
            return jsonify({'error': 'Failed to generate CSV'}), 500
    if __name__ == '__main__':
        app.run(debug=True)
```

4.3.5 RESULT PAGE

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <title>Results</title>
```

```
        <link rel="stylesheet" href="{{ url_for('static', filename='styles1.css') }}">
    </head>
    <body>
        <div class="table-container">
            <h2>Analysis Results</h2>
            <table class="table">
                <thead>
                    <tr>
                        <th>Prediction</th>
                        <th>ATS Score</th>
                        <th>Skills</th>
                        <th>Suggested Job</th>
                        <th>Eligibility</th>
                        <th>Salary Prediction</th>
                        <!-- <th>Email</th>
                        <th>Phone</th> -->
                    </tr>
                </thead>
                <tbody>
                    <tr>
                        <td>{{ prediction }}</td>
                        <td>{{ ats_score }}</td>
                        <td>{{ skills }}</td>
                        <td>{{ suggested_job }}</td>
                        <td>{{ eligibility }}</td>
                        <td>{{ salary_prediction }}</td>
                        <!-- <td>{{ email }}</td>
                        <td>{{ phone }}</td> -->
                    </tr>
                </tbody>
            </table>
            <div class="job-description">
                <h3>Job Description</h3>
                <p>{{ job_description }}</p>
            </div>
            <div class="contact-info">
                <h3>Contact Information</h3>
                <p><strong>Emails:</strong> {{ emails | join(', ') }}</p>
                <p><strong>Phone Numbers:</strong> {{ phones | join(', ') }}</p>
            </div>

            <div class="download-buttons">
```

```
                <a href="{{ url_for('download_pdf') }}">Download PDF Report</a>
                <a href="{{ url_for('download_csv') }}">Download CSV Report</a>
            </div>
        </div>
    </body>
    </html>
```

4.3.4 SCRIPT.JS

```
document.addEventListener('DOMContentLoaded', function() {
    const form = document.getElementById('resume-form');
    const loadingDiv = document.getElementById('loading');
    const errorDiv = document.getElementById('error');


    console.log('DOM fully loaded and parsed');

    if (form) {
        form.onsubmit = function (event) {
            event.preventDefault();
            console.log('Form submitted');

            const formData = new FormData(form);
            console.log('Form data:', formData);

            loadingDiv.style.display = 'block';  // Show loading message
            errorDiv.style.display = 'none';     // Hide error message

            fetch('/predict', {
                method: 'POST',
                body: formData
            })
            .then(response => response.json())
            .then(data => {
                console.log('Response data:', data);
                loadingDiv.style.display = 'none';  // Hide loading message

                if (data.error) {
                    errorDiv.innerText = data.error;
                    errorDiv.style.display = 'block';
                    return;
```

```
                }

                // If on the results page, populate the results
                if (window.location.pathname === '/results') {
                    populateResults(data);
                } else {
                    // Redirect to results page with query parameters
                    const queryParams = new
URLSearchParams(data).toString();
                    console.log('Redirecting to /results with
queryParams:', queryParams);
                    window.location.href = '/results?' + queryParams;
                }
            })
            .catch(error => {
                console.error('Error:', error);
                loadingDiv.style.display = 'none';  // Hide loading
message
                errorDiv.innerText = 'An error occurred: ' +
error.message;
                errorDiv.style.display = 'block';
            });
        };
    }

    // If on the results page, populate the results from query
parameters
    if (window.location.pathname === '/results') {
        populateResultsFromQueryParams();
    }
});

function populateResults(data) {
    document.getElementById('prediction').innerText = data.prediction
|| 'N/A';
    document.getElementById('ats_score').innerText = data.ats_score ||
'N/A';
    document.getElementById('skills').innerText = data.skills ?
data.skills.join(', ') : 'N/A';
    document.getElementById('suggested_job').innerText =
data.suggested_job || 'N/A';
    document.getElementById('job_description').innerText =
data.job_description || 'N/A';
    document.getElementById('eligibility').innerText = data.eligibility
? 'Yes' : 'No';
```

```
    document.getElementById('salary_prediction').innerText =
data.salary_prediction || 'N/A';
    document.getElementById('email').innerText =
data.contact_info.email ? data.contact_info.email.join(', ') : 'N/A';
    document.getElementById('phone').innerText =
data.contact_info.phone ? data.contact_info.phone.join(', ') : 'N/A';
}

function populateResultsFromQueryParams() {
    const urlParams = new URLSearchParams(window.location.search);

    console.log('URL Params:', urlParams.toString());
    document.getElementById('prediction').innerText =
urlParams.get('prediction') || 'N/A';
    document.getElementById('ats_score').innerText =
urlParams.get('ats_score') || 'N/A';
    document.getElementById('skills').innerText =
urlParams.get('skills') ? urlParams.get('skills').split(',').join(', ')
: 'N/A';
    document.getElementById('suggested_job').innerText =
urlParams.get('suggested_job') || 'N/A';
    document.getElementById('job_description').innerText =
urlParams.get('job_description') || 'N/A';
    document.getElementById('eligibility').innerText =
urlParams.get('eligibility') === 'true' ? 'Yes' : 'No';
    document.getElementById('salary_prediction').innerText =
urlParams.get('salary_prediction') || 'N/A';
    document.getElementById('email').innerText = urlParams.get('email')
? urlParams.get('email').split(',').join(', ') : 'N/A';
    document.getElementById('phone').innerText = urlParams.get('phone')
? urlParams.get('phone').split(',').join(', ') : 'N/A';
}

```

4.3.4 STYLE.CSS

/* General Body Styling */

body {

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

margin: 0;

padding: 0;

background: linear-gradient(to right, #00bcd4, #4caf50); /* Gradient background */

```
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
}
/* Centering the Content Container */
.container {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    width: 100%;
    max-width: 800px;
    padding: 20px;
    background-color: #fff;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    opacity: 0; /* Initially hidden for animation */
    animation: fadeIn 1s forwards; /* Fade in animation */
}
/* Navigation Bar Styling */
nav ul {
    display: flex;
    justify-content: center;
    list-style-type: none;
    padding: 0;
    margin: 20px 0;
    background-color: #007bff;
    border: 2px solid #0056b3; /* Border around the nav bar */
    border-radius: 10px; /* Rounded corners for the nav bar */
```

```
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2); /* Shadow for depth */
    animation: slideIn 0.5s ease-out; /* Slide in animation */
}
nav ul li {
    margin: 0;
}
nav ul li a {
    color: white;
    text-decoration: none;
    padding: 15px 20px;
    display: block;
    transition: background-color 0.3s, color 0.3s, transform 0.3s;
    border-radius: 5px; /* Rounded corners for links */
}
nav ul li a:hover {
    background-color: #0056b3;
    color: #fff;
    transform: scale(1.05); /* Slight zoom effect */
}
/* Form Styling */
#resume-form {
    width: 100%;
    border: 2px solid #007bff; /* Border around the form */
    border-radius: 10px; /* Rounded corners for the form */
    padding: 20px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); /* Shadow for depth */
    animation: fadeInUp 1s ease-out; /* Fade in-up animation */
    background-color: #f9f9f9; /* Light background for form */
}
.form-group {
```

```
    margin-bottom: 20px;
}
.form-group label {
    display: block;
    margin-bottom: 8px;
    color: #333;
    font-weight: bold;
    font-size: 16px;
}
.form-group input[type="file"],
.form-group input[type="text"] {
    width: calc(100% - 20px); /* Full-width input with padding
adjustment */
    padding: 10px;
    border: 2px solid #ddd; /* Border for input fields */
    border-radius: 5px;
    box-sizing: border-box;
    transition: border-color 0.3s ease-in-out;
}
.form-group input[type="file"]:focus,
.form-group input[type="text"]:focus {
    border-color: #007bff; /* Focus border color */
}
.form-group button {
    width: 100%;
    padding: 10px;
    background-color: #007bff;
    color: #fff;
    border: none;
    border-radius: 5px;
```

```
    cursor: pointer;
    font-size: 16px;
    transition: background-color 0.3s, transform 0.3s;
}
.form-group button:hover {
    background-color: #0056b3;
    transform: scale(1.05); /* Slight zoom effect */
}
/* Animations */
@keyframes fadeIn {
    from {
        opacity: 0;
        transform: translateY(20px); /* Move content from below */
    }
    to {
        opacity: 1;
        transform: translateY(0);
    }
}
@keyframes fadeInUp {
    from {
        opacity: 0;
        transform: translateY(20px); /* Move content from below */
    }
    to {
        opacity: 1;
        transform: translateY(0);
    }
}
@keyframes slideIn {
```

```
    from {
        transform: translateY(-20px); /* Start from above */
        opacity: 0;
    }
    to {
        transform: translateY(0);
        opacity: 1;
    }
}
```

CHAPTER 5

SOFTWARE TESTING

Software testing is crucial to ensuring the correctness, completeness, security, and quality of developed computer software. Testing is a process of investigation, not just following a procedure, and it involves questioning a product to evaluate it. The quality attributes of software can vary widely, including reliability, stability, portability, maintainability, and usability. Testing should include assessing the system's ability to handle various resume formats and styles, its effectiveness in parsing and interpreting content, and its alignment with the organization's hiring criteria.

5.1 Testing Strategies

Designing effective test cases is essential, as is the strategy for executing them. A systematic strategy for testing software helps avoid wasting time and effort. In the context of a music concert management system, the following testing strategies can be applied:

1. **UNIT TESTING**

   Unit testing focuses on verifying the smallest units of software, such as software components or modules. For a music concert management system, unit testing could involve testing the functionality of the artist management module, ensuring that artist information is displayed correctly, and that the system handles songs details and concert dates viewing accurately.

2. **INTEGRATION TESTING**

   Integration testing involves constructing the program structure while simultaneously conducting tests to uncover errors associated with interfacing. In a music concert management system, integration testing could involve testing how the artist management module integrates with the song details and concert dates viewing module to ensure seamless interaction between the components.

3. **VALIDATION TESTING**

Validation testing is conducted to ensure that the software functions as expected by the customer. Validation can be defined in mainly ways but a simple definition is that validation succeeds when software functions in a way that can be reasonably expected by a customer.

4. **SYSTEM TESTING**

System testing involves fully exercising the computer-based system. For a music concert management system, system testing could include testing the system's ability to handle various scenarios.

5.2 Test Cases

Table 5.1 Test Case Specifications

TEST CASE NO	TEST CASES	INPUT DATA	EXPECTED OUTPUT	OBTAINED OUTPUT
1	Upload resume file	Resume File (Pdf, doc, docx)	Resume file should be analyzed.	Analysed and displayed
2	Display analysis result	Valid Resume	Result should be successfully displayed.	Result should successfully submit.
3	Searching for Jobs	Entering keywords or job type in search	Relevant job listings to be displayed	Relevant job listings to be displayed
4	Download pdf, csv	Valid resume file (pdf, csv)	Should be downloaded successfully	Downloaded successfully.

CHAPTER 6

RESULTS

Snapshot-01:

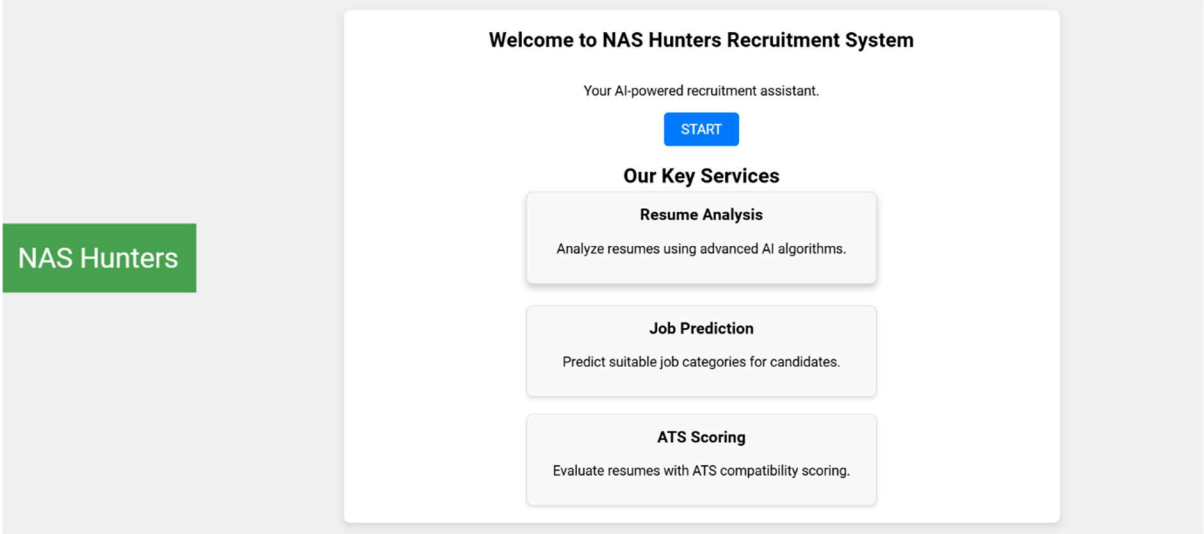

Figure 6.1 Homepage

Figure 6.1The NAS Hunters Recruitment System offers AI-powered services, including Resume Analysis, Job Prediction, and ATS Scoring.

Snapshot-02:

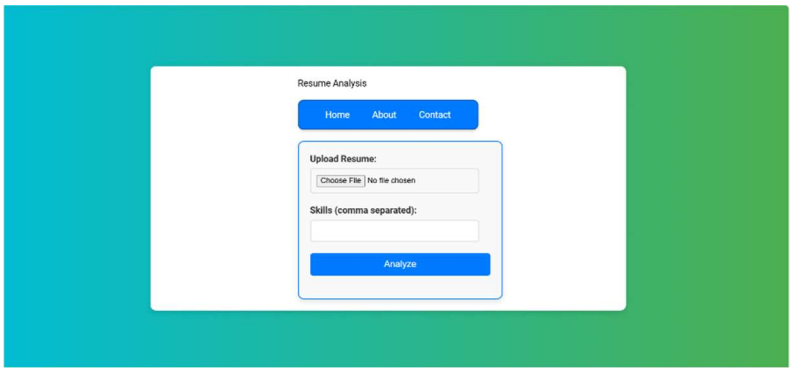

Figure 6.2 Resume Analysis Page

Figure 6.2 The Resume Analysis page allows users to upload their resume and input relevant skills for analysis. The page also includes navigation buttons

Snapshot-03:

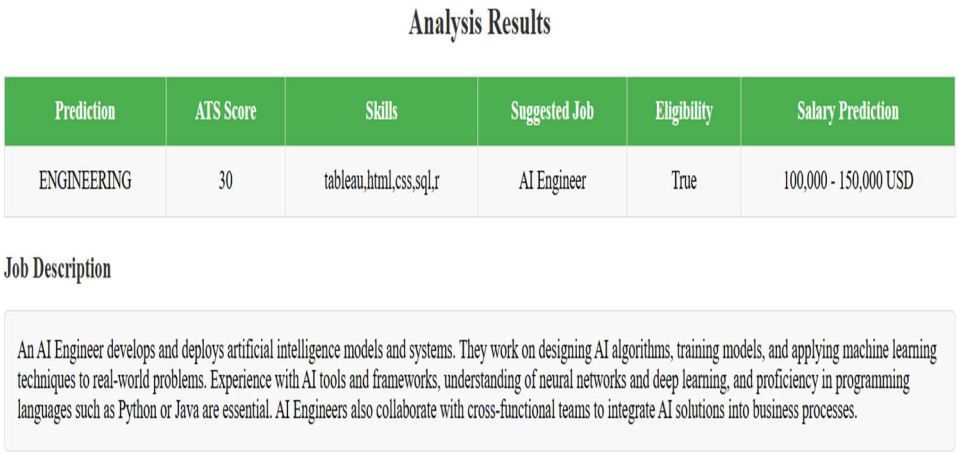

Figure 6.3 The Analysis Results page

Figure 6.3 displays the predicted job category, ATS score, relevant skills, suggested job title, eligibility, and salary prediction. It also includes a detailed job description

Snapshot-04:

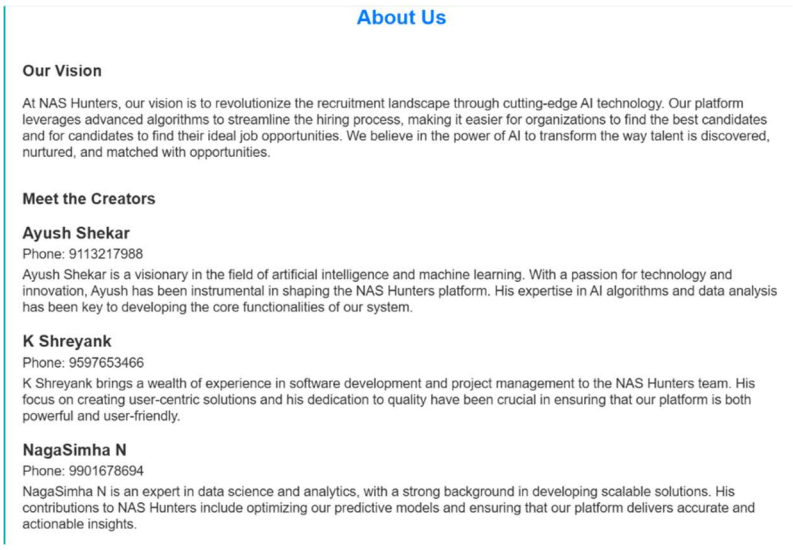

Figure 6.4 About Us Page

Figure 6.4 contains the "About Us" section for NAS Hunters, which includes the company's vision and information about its creators.

Snapshot-05:

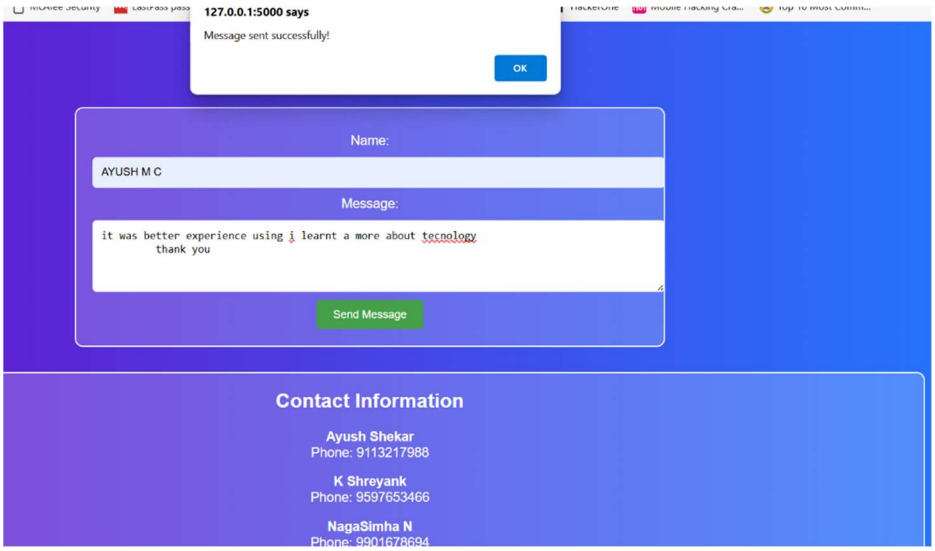

Figure 6.5 Contact Us Page

Figure 6. For any inquiries, please reach out to our team using the contact information provided below. We're here to assist you with any questions

CONCLUSION & FUTURE ENHANCEMENT

CONCLUSION

AI Recruitment System serves as an efficient and user-friendly platform designed to streamline the job application and recruitment process. By offering distinct dashboards for employers and employees, the system ensures that both parties can effectively manage job postings and applications. The Employer Dashboard provides robust tools for posting jobs, reviewing applications, and analyzing recruitment metrics, facilitating better hiring decisions for the HR.

Meanwhile, the Employee Dashboard offers a seamless experience for job seekers to browse opportunities, apply for positions, and track their application progress. The Admin Page enhances overall system management by allowing administrators to oversee user activity, manage job postings, and ensure smooth operation.

FUTURE ENHANCEMENT

Future enhancements to the AI Recruitment System could significantly elevate its functionality and user experience. Integrating advanced AI algorithms for personalized job recommendations would enhance job matching accuracy, while implementing machine learning for resume parsing and automated candidate screening would streamline recruitment processes.

Expanding the platform to support multi-language interfaces and accessibility features would make Job Connect more inclusive. Enhanced data analytics tools could provide recruiters with actionable insights into hiring trends and candidate behaviors.

Additionally, integrating with social media platforms and professional networks could widen job reach and facilitate easier profile management. A mobile app version of the platform would further improve accessibility and user engagement, enabling users to manage their job search and recruitment activities on the go.

REFERENCES

[1] Faliagka, Evanthia, et al. "Application of machine learning algorithms to an online recruitment system." Proc. International Conference on Internet and Web Applications and Services. 2023

[2] Anagha Prakash, Rajiv Nair., 2022. International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-9 Issue-Perception of Fresh Graduates towards Job Portal Sites.

[3] Sumi Maharjan., 2022. Quest Journal of Management and Social Sciences Volume 1 Issue 2: 308-317 ISSN Print: 2705-452. Graduates Perception of Job Search: A Critical Review.

[4] Kelley, Erin M., Christopher Ksoll, and Jeremy Magruder. "How do Online Job Portals affect Employment and Job Search? Evidence from India." (2022)

[5] Karthik R., 2022. International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-7, Issue-6. A study on improving the marketability of job-related services among the recruiters with reference to online job portal.

[6] Mansi Srivastava, Shweta Singh, Department of Computer Science and Engineering School of Management Sciences Aff. APJ Abdul Kalam Technical University, Lucknow, Uttar Pradesh, India Volume: 05/Issue: 05/May-2022 Impact Factor- 7.868 e-ISSN: 2582-5208

[7] ©2022, SIJTRD, Shrimathi Devkunvar Nanalal Bhatt College for women, all rights reserved http://www.sdnbvc.edu.in/sijtrd

[8] Keethana Kopuri1, Gulam Mujtaba Hussain Aqueel2, Azbar Sadiqa Jabeen3, Dr.T.K. Shaik Shavali4 B. Tech student, Dept. of AIML, Lords Institute of Engineering and Technology February 2021 IJIRT Volume 3 Issue 9 ISSN: 2349-6002

BIBLIOGRAPHY

WEB REFERENCES

- http://www.google.com
- https://getbootstrap.com/
- https://www.djangoproject.com/
- www.wikipedia.org

BOOKS REFERENCES

- HTML & CSS by Thomas A. Powell, Fifth Edition.
- Web Design by Jennifer N. Robbins, Fifth Edition.
- Data Base Management System by Ramakrishnan and Gehrke, Third Edition.
- Django for Beginners by William S. Vincent